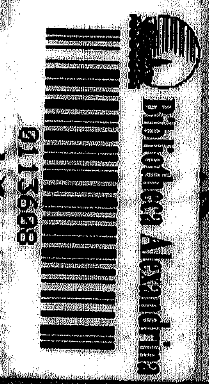
١٩
سلسلة تيسير علوم الحاسب
مجموعة كتب برامج الحاسب

مجدي محمد أبو العطا

# المرجع الأساسي لمستخدمي C






محمد عزب



كمبيوساينس
العربية لعلوم الحاسب

المرجع الأساسى لمستخدمى

# C - Language

تأليف : عزب محمد عزب

مراجعة : مجدى محمد أبو العطا

الطبعة الأولى

١٤١٦هـ - ١٩٩٦م



كمبيوساينس

العربية لعلوم الحاسب

المركز الرئيسى : ٤٩ شارع الحجاز . أمام دار المناسبات - مصر الجديدة. القاهرة

ت / فاكس : ٢٤٠٥٣٣٠ - ٢٤٩١٢٩٥

حقوق الطبع محفوظة للناشر ، ولا يجوز نشر أي جزء من هذا الكتاب أو اعادة طبعة أو تصويره أو اختزان مادته العلمية بأية صورة دون موافقة كتابية من الناشر.

رقم الايداع : ٩٦/١٠٠٦٤

I. S. B. N : 977-5735-03-3

# الكتاب فى سطور

يشتمل كتاب المرجع الأساسى لمستخدمى C على خمسة عشر فصلاً نوجزها فيما يلى

الفصل الأول.. نظرة عامة تشمل مقدمة تاريخية عن لغة C ومميزاتها والخطوات اللازمة لكتابة وتنفيذ برنامج بلغة C والقواعد التى يجب اتباعها وأخيراً أنواع البيانات والمؤثرات.

الفصل الثانى.... دوال الإدخال والإخراج والتى تسمح بقبول البيانات وإظهارها وتغيير ألوانها وطباعتها والتعامل مع الطابعة.

الفصل الثالث... الدوارات وأنواعها والفرق بينها ومتى تستخدم واحدة دون الأخرى ، واستخدام الدوارات اللانهائية.

الفصل الرابع... التفريع المشروط وغير المشروط والتراكيب المختلفة لجملة IF.

الفصل الخامس... كيفية إنشاء الدوال والماكرو والفرق بينهما مع اعطاء أمثلة بسيطة.

الفصل السادس... المصفوفات ذات البعد وذات البعدين وكيفية الاعلان عنها والتعامل معها وطباعة عناصرها ، وإرسال المصفوفات إلى دالة كعامل ، وأخيراً أوامر المترجم.

الفصل السابع... توظيف المفاتيح والتحكم فى حركة المؤشر والتعريف بملف ANSI.SYS واستخداماته ومتطلباته والتحكم فى خصائص الحروف وكيفية تصميم قوائم الإختيارات.

الفصل الثامن... مؤشرات العناوين (Pointers) وضرورة إستخدامها وكيفية تمرير مصفوفات إلى الدوال بإستعمال المؤشرات ، والعلاقة بين المؤشرات والعبارات الحرفية وأخيراً مصفوفة الحرفيات والمؤشرات.

الفصل التاسع... السجلات Structures يشـرح معنى السـجل والحاجـه إلى اسـتعماله و كيفية انشائه والتعامل مع عناصره ، واستعمال السجل كعنصر في سـجل آخـر والتعـامل مع مصفوفة السجلات وكيفية الاعلان عن مؤشر من نوع Structure.

الفصل العاشر... الملفات ويتناول الطرق المختلفة للتعامل مع الملفات ودوال فتح الملف وغلقه وكيفية القراءة والكتابة حرف بحرف أو عبارة فى المرة أو سجل فى المرة أو مجموعة بيانات.

الفصل الحادى عشر... يشرح دوال مهمة للمبرمج تساعد فى نسخ جزء من الشاشة إلى موضع آخر وحفظه واسترجاعه ، والتعامل مع الألوان والصوت بالإضافة إلى برنامج شامل يستخدم كمحرر للنصوص.

الفصل الثانى عشر... الرسم فى لغه C ويشرح تهيئة الشاشه لحالة الرسم ، وكيفية رسم الخطوط بأنواعها مع تغيير الألوان ، ورسم القطع الناقص ومتعدد الخطوط وكيفية تلوين الرسم أو تظليله وكيفية الحصول على رسم بيانى ذو بعدين أو ثلاثى الأبعاد وتحريك الرسوم واستخدام الخطوط فى عمل أشكال رسومية.

الفصل الثالث عشر... يشرح دوال التعامل مع الذاكرة مثل حجز مساحة من الذاكرة ، وتفريغها وإعادة تخصيصها وحجز مساحة متغيرة.

الفصل الرابع عشر... يشرح روتينات الذاكرة Rom Bios وكيفية استعمالها مع شرح لبرامج تحديد حجم الذاكرة والتعامل مع المؤشر.

الفصل الخامس عشر... عبارة عن مثال شامل يشتمل على معظم الموضوعات التى مرت بالفصول السابقة. وهذا البرنامج يمكن الاستفادة منه فى إعداد برامج مماثلة تستفيد منها فى حياتك العملية.

الملاحق... الملحق الأول.شفرة تبادل المعلومات ASCII ،الملحق الثانى التعامل مع محرر البرامج ، الملحق الثالث الكلمات المحجوزة فى لغة C

# المحتويات

الفصل العاشر



الفصل الحادى عشر

# تقديم

ان الحمد لله ، نحمده ونستعينه ونستهديه ، ونصلى ونسلم على سيدنا محمد صلى الله عليه وسلم وآله وصحبه أجمعين .

﴿ سبحانك لاعلم لنا الا ماعلمتنا ، انك أنت العليم الحكيم ﴾ ... وبعد

بفضل من الله كان النجاح الذى حققته سلسلة تيسير علوم الحاسب وبقدر الجهد الذى بذل كان النجاح الذى تحقق . وكانت سعادتى غامرة بالرضا الذى استقبل به القراء كتبنا . لقد كان التشجيع من الجميع دافعا وحافزاً كما كان النقد البناء والملاحظات القيمة من الزملاء ومن كل من قرأ كتب سلسلتنا – والذى عبرت عنه رسائلهم إلينا – كنزاً نعتز به

وهذا الكتاب اضافة جديدة إلى سلسلة تيسير علوم الحاسب التى نتشرف باصدارها . والكتاب يشتمل على معلومات وافية وغزيرة تحرَّى فيه المؤلف الدقة التامة ، وعلى كم هائل من البرامج الهدف منها تحقيق أقصى فائدة ولتكون بمثابة نماذج يقتدى بها فى اعداد برامج مماثلة فى حياتك العملية .

ان طموحاتنا كبيرة وآمالنا عظيمة وثقتنا بالله بغير حدود. ولذلك فاننا نناشد كل صاحب قلم أو فكر من المهتمين والمتخصصين في مجال الحاسب أن يضع يده في أيدينا لكى نخرج أفكارة ومؤلفاته إلى النور ونساهم في سد العجز الهائل في هذا المجال .

مجدى محمد أبو العطا

# مقدمة

اذا كنت قد تعودت على أن تتعامل مع برنامج نوافذى تشعر معه أنك مقيد الحركات، جامد الإبداع فسيتحول هذا الشعور إلى الأفضل من خلال دراسة أقوى لغات البرمجة للمحترفين على الإطلاق.

لقد حاولنا في هذا الكتاب أن نزيل الحاجز النفسي بين عامة المستخدمين ولغات البرمجة وخاصة لغة C، ولا نطلب منك إلا أن تعطينا الفرصة لإتمام ذلك وتشاركنا عملية تحويل المستخدم إلى مبرمج يصبح غيرة مستخدماً لما ينتجه.

فأن تكون منشيء الشيء ، لا من يستخدمه ...هذا ما نرتضيه لك

أن تكتشف أغنى ملكاتك ... هذا ما سنساعدك عليه

أن تقف موقف الملقى فى ساحة العطاء...هذا ما نصبو إليه ·

م/ عزب محمد عزب

# الفصل الأول
# نظرة عامة

نتناول فى هذا الفصل نظرة عامة تشمل التعريف بلغة C وموقعها بين لغات البرمجة الأخرى ومميزاتها. وكيفية كتابة برنامج بلغة C لأول مرة وأنواع البيانات والمؤثرات التى تستخدمها C.

انتهاء هذا الفصل ستتعرف على :

- مقدمة تاريخية عن لغة C.
- مميزات لغة C.
- فتح ملف البرنامج.
- خطوات كتابة وتنفيذ برنامج بلغة C
- قواعد كتابة البرنامج.
- شرح مكونات أول برنامج.
- أنواع البيانات.
- أنواع المؤثرات (Operators).

فــي عــام ١٩٦٠ صممــت لغــة Combined Programming CPL Language في جامعة لندن وقد استخدمت فيها بعض تعليمات لغة ALGOL-60 ، وفي عام ١٩٦٧ انبثقت لغة BCPL من لغة CPL على يد مارتن ريتشاردز ، ثم قام ثومبسـون بتطوير BCPL وسماها لغة B وكل هـذه اللغـات تعتبر مـن لغـات المسـتوى الأدنـى (Low Level Languages) أي القريب من لغة الآلة مثل لغة التجميع (Assembly).

وفي عام ١٩٧٢ وفي معامل شركة AT&T الأمريكية قام ريتشي بإستباط لغة جديدة مـن لغة B أخذ منها أحسن تعليماتها وأضـاف إليهـا أوامـراً جديـدة وأنواعـاً جديـدة للبيانـات وكثيرا من الدوال التي تفيد المبرمج وسميت هـذه اللغـة بلغـة C. ومنـذ ذلـك التـاريخ أخذت لغة C شهرة واسعة لأنها أصبحت تنتمي للغات المستوى الأعلى (High Level Languages) مثل لغة البيسيك والباسكال من حيث سهولة الإستخدام مـن ناحيـة ومـن ناحية اخرى تنتمي الى لغات المستوى الأدنى من حيث قدرة اللغة على مخاطبة مكونـات الجهاز (Hardware)

ومع نجاح وانتشار لغـة C انتشـرت لهجـات كثيـرة للمتحدثيـن بلغـة C وكـاد يحدث معها ما حدث للغة بيسيك من وجود بيسيك خاص لكل نوع مـن أنـواع الأجهـزة سواء كانت أجهزة أي بي ام أو أبل أوكومودور أو سنكلير لولا أن قـام معهـد القياسـات الأمريكية بعملية توحيد لهذه اللهجات المختلفة فـأصدر فـي عـام ١٩٨٣ اللغـة القياسـية ANSI C ، والجديـر بـالذكر أن معظـم مترجمـات لغـة C تتوافـق مـع ANSI C مثـل Borland C, Microsoft C, Lattice C.

## مميزات لغة C

نورد فيما يلي أهم ما يميز لغة C عن غيرها من لغات البرمجة وهـي المميـزات التـي تدعـوك لتفضيلهـا علـى غيرهـا مـن لغـات البرمجـة المعروفـة وترغبـك فـي تعلمهـا واستخدامها.

تتميز لغة C بمجموعة من المزايا مثل:

*لغة عـامة*

أى تصلح لعمل برامج قواعد البيانات والرسومات والحسـابات ونظـم التشـغيل وغيرها

***لغة تركيبية (Structured Language)***

البرنامج المكتوب بلغة C عبارة عن دالـة رئيسـية تنـادى مجموعـة مـن الـدوال الأخرى وكل دالة مجموعة من الأوامر.

***تتعامل على مستوى "البت" (Bit Manipulation)***

حيث تستطيع أن تقرأ وتكتب وتغير وتقوم بعمليات على مستوى الـ Bit وكمـا هو معروف فإن الـ Bit هى أصغر وحدة لقياس المعلومات داخل الكمبيوتر وهى جزء من ثمانية أجزاء تعادل فى مجموعها حرف واحد (Byte).

وهذه الميزة جعلتهـا متخصصـة فى بعـض مجـالات التحكـم الآلـى والروبـوت وبرامج الـ Utility وبرامج معالجة الصور وضغط الملفات واكتشاف الأعطال.

***لغة متنقلة (Portable)***

أى يمكن للبرنامج المكتوب بلغـة C أن يعمـل مـع اكـثر مـن جهـاز مثـل IBM Apple أو الأجهزة المتوسطة والكبيرة مع بعض التعديلات الطفيفة.

*لغة سريعة*

لأن أدوات اللغة تتعامل مباشرة مع الآلة مما يختصر وقت التنفيذ.

*لغة قياسية*

معظم مترجمات اللغة تتوافق مع اللغة القياسية ANSI C.

*لغة نظام التشغيل Unix*

مما يدل على ثراء هذه اللغة ومرونتها أن نظام التشغيل المشهور UNIX مكتوب بها.

## تثبيت قرص التمارين المرفق بالكتاب

تجد مع هذا الكتاب قرص مرفق باسم : المرجع الاساسى للغة C .. التمارين العملية. يشتمل القرص على التمارين العملية الموجودة بالكتاب والتى يعتمد عليها الشرح الوارد بالكتاب. ويفضل نسخ التمارين الموجودة علىالقرص إلى القرص الصلب قبل أن تشرع فى دراسة هذا الكتاب.

فيما يلى خطوات تثبيت القرص المرفق مع الكتاب :

١. قم بعمل نسخة احتياطية من هذا القرص باستخدام أمر DISKCOPY ثم احفظ النسخة الأصلية من القرص فى مكان آمن.

٢. ضع القرص المنسوخ فى المشغل A أو B حسب ماهو متوفر فى جهازك

٣. أكتب من مشغل القرص الموجود به الاسطوانة الامر Install بالصورة التالية :

```
A:\> INSTALL
```

سيقوم برنامج التثبيت(INSTALL) بنسخ محتويات القرص (جميع أمثلة الكتاب) الى القرص الصلب تحت الدليل C_BOOK ويقف المحث داخل هذا الدليل

٤. أكتب الامرDIR

ستجد أدلة فرعية بأسماء فصول الكتاب وكل دليل فرعي يحتوى على تمارين الفصل فمثلا الملف CS_1-1.C معناه المثال الاول فى الفصل الاول حيث CS اختصارا لاسم الشركة Compu Science و ١-١ تعنى الشكل الأول من الفصل الأول.

## تشغيل البرنامج لأول مرة

نفترض ان لديك جهاز مناسب للعمل مع لغة C كما أشرنا من قبل وأنك قمت بتثبيت قرص التمارين على القرص الصلب (Hard Disk) وأيضا قمت بتثبيت أحد البرامج المترجمة للغة C (C Compiler) ولتكن Turbo C أو Turbo C++ أكتب أمر تشغيل البيئة المتكاملة IDE

بفرض أن الدليل الخاص ب Turbo C أو Turbo C++ هو tc تابع الخطوات التالية :

١. اكتب الأمر التالى ثم إضغط مفتاح الإدخال

```
cd \ tc
```

يظهر المحث بهذا الشكل

```
c:\tc>
```

٢. اكتب الأمر التالى ثم إضغط مفتاح الإدخال

```
cd bin
```

يظهر المحث بهذا الشكل

```
c:\tc\bin>
```

٣. من محث DOS أكتب الأمر التالى ثم إضغط مفتاح الإدخال

```
c:\tc\bin>tc
```

تظهر الشاشة الخاصة بتحرير البرامج كما فى شكل (١-١)

≡ File Edit Search Run Compile Debug Project Options

شكل (١-١) شاشة تحرير البرامج

تستخدم هذه الشاشة لكتابة البرامج وتعديلها وترجمتها وإكتشاف الأخطاء وأيضا تنفيذها.

وتتكون الشاشة من مجموعة من القوائم المنسدلة الخاصة بالملفات (File) وتحرير السطور (Edit) والبحث عن الكلمات وإستبدالها (Search) وترجمة البرامج (Compile) وتنفيذها (Run) والبحث عن الأخطاء وتصحيحها (Debug) وبعض الخيارات الأخرى (Options).

لتنشيط شريط القوائم اضغط مفتاح Alt ولفتح قائمة اضغط مفتاح Alt مع أول حرف من إسم القائمة مثلا لفتح قائمة File اضغط Alt+F.



راجع الملحق (ب) للحصول على تفصيلات عن كيفية التعامل مع برنامج Turbo C.

للخروج من البرنامج ثانية إلى محث DOS

٣. اضغط Alt+F تظهر قائمة File

٤. اختر Exit أو اضغط المفتاحين Alt+X معاً

## فتح ملف البرنامج

إذا كانت لك خبرة بلغة بيسيك فإنك تعرف أنه بإستطاعتك أن تكتب برنامج وتنفذه فى أول ساعة وسنثبت لك أنك تستطيع أن تفعل ذلك مع لغة C أيضا.

افتح بيئة تشغيل البرامج من جديد بالأمر TC بالصورة التالية :

c:\tc\bin>tc

افتح ملف البرنامج من دليل التمارين بإتباع الخطوات الآتية :

١. اضغط Alt+F

تظهر قائمة أوامر التعامل مع الملفات

٢. اختر الأمر Open

يظهر مربع حوار فتح الملفات (شكل ٢-١)

فى خانة Name اكتب إسم الملف cs1_1 إسم الدليل كالآتى :

```
C:\c_book\cs1_3.c        Open
```

شكل ٢-١

٤. اضغط Enter

يظهر البرنامج كما فى الشكل ٣-١

```
/* Program Name ; CS1_3.C */
/* This program prints the message wellcome with CompuSience */
#include <stdio.h>
main()
```

```
{
  printf ("Welcome With CompuSience ");   /*output function */
}
```

شكل رقم ٣-١ شكل البرنامج بعد كتابته

و الأفضل أن تكتب البرنامج للتعرف على أوامر اللغة والشعور بالألفة معها. قبل ان نشرح البرنامج والتعليمات التى يشتمل عليها سنشرح قواعد كتابة البرنامج وخطوات تنفيذه بصفة عامة وعليك ان تطبقها على البرنامج الذى بين ايدينا.

## قواعد بناء البرنامج

البرنامج cs1_1.c الموجود بشكل ٣-١ يمثل أبسط تركيب لبرنامج مكتوب بلغة C ويخضع للقواعد الآتية :

- يبدأ البرنامج بالعبارة #include <........h> وبين العلامتين إسم ملف التوجيه الخاص بالدوال المستخدمة فى البرنامج (سيأتى ذكر ملف التوجيه الخاص بكل دالة عند شرحها)
- يتكون البرنامج من دالة رئيسية main() وتبدأ بالقوس } وتنتهى بالقوس {
- جميع كلمات ودوال اللغة تكتب بالحروف الصغيرة
- تنتهى كل عبارة بفاصلة منقوطة (;) مع وجود استثناءات سوف تعرفها فيما بعد
- يجوز كتابة أى ملاحظات أو تعليقات خاصة بالمبرمج بوضعها بين العلامتين */ /* لأى عدد من السطور.

## التعامل مع البرنامج

فيما يلى نوضح كيفية التعامل مع البرنامج :

- لحفظ ملف البرنامج اضغط F2 أو إختر Save من قائمة File
- لترجمة وتنفيذ البرنامج اضغط على المفتاحين Ctrl+F9 أو ALT+R ثم على الاختيار الاول من القائمة نضغط مفتاح الادخال
- لرؤية نتيجة التنفيذ اضغط على المفتاحين Alt+F5 تظهر النتيجة كما يلى :

**Welcome With CompuSience**

- للرجوع الى شاشة الكتابة اضغط أى مفتاح

## خطوات التنفيذ

لاحظ أن البرنامج الذى تم تنفيذه هو cs1_1.exe وليس cs1_1.c حيث تمت ترجمة البرنامج المصدر C. وتحويله الى ملف قابل للتنفيذ (executable) على الخطوات الآتية :

- ضم ملف (ملفات) التوجيه المعلن عنه بالعبارة #include <stdio.h> إلى الملف المصدر cs1_1.c
- ترجمة الكود بعد الضم إلى لغة الآلة ليعطى ملف له الامتداد obj.
- ضم ملفات مكتبات الدوال إلى الملف الناتج وربطهم بواسطة برنامج Linker فى ملف واحد وهو cs1_1.exe والملف s1_1.exe لايحتاج لبيئة تشغيل خاصة بل مثل أى برنامج تحت محث DOS ولتشغيل البرنامج أخرج من برنامج Turbo C بالضغط على Alt+x واكتب إسم البرنامج بالصورة التالية :

**c:\tc\bin>cs1_1**

والان نعود لشرح البرنامج الموجود بالشكل ٣-١

## شرح البرنامج

- السطور رقم ١ ، ٢ ، ٣ عبارة عن تعليقات (comments) لاتؤثر فى تنفيذ البرنامج ويتم ذلك بوضع */ فى بداية التعليقات ووضع /* فى نهاية التعليقات مهما كان عدد سطور التعليقات.

- السطر رقم ٤ هو <stdio.h> #include فيه كلمة #include بمعنى اشمل وكلمة stdio.h هىملف اسمه stdio وإمتداده h هذا الملف موجود مع حزمة برنامج لغة C وذلك فى الدليل INCLUDE والسطر كله معناه حمل (أضف محتويات هذا الملف الى البرنامج) الملف stdio.h لأن به تعريفات الدوال التى سوف تستخدم فى البرنامج ومن هنا نفهم أن كل مجموعة دوال من دوال لغة Cلها ملف مثل الملف stdio.h يجب كتابته فى أول البرنامج وسوف نتعرض لهذه النقطة بالتفصيل فى موضع أخر.

- السطر رقم ٥ يحتوى على عبارة ()main هذه العبارة هى الدالة الرئيسية للبرنامج حيث أن البرنامج سوف يتركب من مجموعة دوال ولكن الدالة ()main هى الدالة الرئيسية التى تنادى باقى الدوال.

- السطر رقم ٦ هو بداية الدالة ()main و تبدأ بالقوس } وتنتهى الدالة الرئيسية فى السطر رقم ٨ بالقوس { بمعنى أننا نفتح القوس } فى أول الدالة الرئيسية وننهى الدالة الرئيسية بالقوس{

- ويشتمل السطر رقم ٧ على العبارة printf ("Welcome With CompuScience. "); وفيها الدالة ()printf وهى دالة الاخراج الرئيسية على الشاشة وسوف نناقشها بالتفصيل فى الفصل الثانى و هنا تطبع عبارة على الشاشة.

- السطر رقم ٨ والأخير ويشتمل على القوس } ومعناه نهاية الدالة الرئيسة .main()

## أنواع البيانات

كما تعرف أن البيانات التى نتعامل معها إما أرقام أو حروف أو كلمات و الارقام يمكن أن تكون صحيحة (أى ليس بها علامة عشرية) أو حقيقية أى بها علامة عشرية.

والحروف يمكن أن تكون حرف واحد أو اكثر من حرف وهكذا تختلف أنواع البيانات عن بعضها البعض و من الضرورى معرفة أنواع البيانات ومعرفة كيفية الاعلان عنها وكذلك كيفية استعمالها.

والجدول الاتى يوضح هذه الانواع وكذلك عدد البايت (Byte) التى يشغلها كل نوع

| نوع المتغير | طوله بالبايت | المدى المسموح |
|---|---|---|
| حرف (char) | ١ | حرف أو رمز واحد |
| صحيح قصير (int) | ٢ | -٣٢٧٦٨ إلى ٣٢٧٦٨ |
| صحيح طويل (long) | ٤ | -٢٠١٤٧٠٤٨٣٠٦٤٨ إلى ٢٠١٤٧٠٤٨٣٠٦٤٨ |
| حقيقى (float) | ٤ | E-38 الى E+38 |
| حقيقى مضاعف (double) | ٨ | E-308 الى E+308 |

ونوضح فيما يلى المقصود بكل هذه الانواع :

- متغير من نوع حرف أى متغير يصلح لتخزين حرف فقط
- متغير من نوع صحيح أى متغير يصلح لتخزين رقم صحيح ( ليس به علامة عشرية مثل ٥ ،٥٧، ٥٤٤ ).

- متغير من نوع صحيح ولكن طويل ( long) اى يستطيع أن يخزن رقم صحيح ضعف المتغير الصحيح العادى ونستعمل هذا النوع اذا كانت الارقام التى نتعامل معها أكبر من المساحة المخصصة للرقم الصحيح العادى والا سنحصل على نتائج خاطئة بالرغم من ان البرنامج سليم .
- متغير حقيقى أى متغير يصلح لتخزين رقم حقيقى ( يقبل الكسور العشر ية مثل ٣,٣ ٤٤,٥ ٣٣,٤ )
- متغير حقيقى مضاعف اى يستطيع أن يخزن رقم حقيقى ضعف المتغير الحقيقى العادى

## تسمية المتغير

- يخضع اسم المتغير لشروط معينة يجب أن تعرفها تجنبا لأخطاء قد تقع فيها وفيما يلى نوضح هذه الشروط :
- يجب ان يبدأ المتغير بحرف ثم يكمل المتغير بعد ذلك حروف أو أرقام ويجب ألا يحتوى على علامة خاصة سوى الشرطة التحتية (_).
- من الممكن أن يشتمل إسم المتغير حتى ٣٢ حرف وما زاد عن ذلك لا يلتفت إليه مترجم اللغة .
- يفرق المترجم بين الحروف الصغيرة و الكبيرة فالمتغير St يختلف عن المتغير st فاذا استعملا فى البرنامج يعتبرهما البرنامج متغيرين
- يجب ألا يكون المتغير باسم كلمة من الكلمات المحجوزة فى اللغة مثل int,return.

## الإعلان عن المتغيرات

اذا كنت تستخدم مترجم للغة C/C++ فيمكن يتم الإعلان عن المتغيرات فى أى مكان بالبرنامج ولكن بشرط ان تكون قبل العبارات التى تستخدم هذا المتغير أما اذا

كنت تستخدم مترجم للغة C فقط فيجب أن يكون الإعلان فى أول البرنامج لتلافى الأخطاء.

*مثال :*

```
int a;
float b;
```

## المؤثرات Operators

المؤثرات هى الرموزالتى تربط بين المتغيرات والثوابت لإنشاء علاقة ما أو معادلة تختلف أنواع المؤثرات بإختلاف وظيفه كل مؤثر. وتأخذ الأنواع الآتية

المؤثرات الحسابية Arithmatic Operators

وهى علامات الجمع (+) والطرح(−) والقسمة(/) والضرب (*) وتستخدم مع المتغيرات والثوابت الرقمية

*مؤثرات المقارنة (Relational Operators)*

وتستخدم لمقارنة قيمتين لمعرفة هل هما متساويتين أو احداهما أكبر أو أقل من الأخرى وهكذا .و يوضح الجدول التالى مؤثرات المقارنة والرموز التى تستخدم بدلا عنها.

| المؤثر | الرمز | مثال | النتيجة |
|---|---|---|---|
| أكبر من | > | 10>8 | 1 |
| اصغر من | < | 10<8 | 0 |
| يساوى | == | 10 ==8 | 0 |
| لايساوى | != | 10!=8 | 1 |
| أقل من أويساوى | <= | 10<=8 | 0 |
| أكبر من أويساوى | >= | 10>=8 | 1 |

## المؤثرات المنطقية Logical Operators

وتستخدم لتحديد شرط مركب مثل الشرط التالى if(a==b && c==d)

ومعناه إذا كانت قيمة المتغير A تساوى قيمة المتغير B وفى نفس الوقت قيمة المتغير C تساوى قيمة المتغير D.

و يوضح الجدول التالى هذه المؤثرات والرموز التى تستخدم بدلا منها :

| المؤثر | الرمز | مثال | النتيجة |
|---|---|---|---|
| و AND | && | 10>8 && 9>7 | 1 |
| أو OR | \|\| | 10<8 \|\| 7<8 | 1 |
| لا NOT | ! | !(10 ==8) | 1 |

## مؤثرات التخصيص Assignment Operators

وهى مؤثرات تخزين قيمة فى متغير مثل =/, =*, =-, = + ,=

وتستخدم لتخزين قيمة فى متغير بالإعتماد على القيمة الموجودة فى نفس المتغير فمثلا إذا قمت بتخزين القيمة ٦ فى المتغير a بإستخدام الأمر a=6 وأردت مضاعفة القيمة المخزنة يجب أن تكتب الأمر a=a*2 بهذا تصبح قيمة a تساوى ١٢ولزيادة قيمة المتغير a=a+1 وهكذا وهذه الطريقة تستخدم فى جميع لغات البرمجة وتتميز لغة C بوجود طريقة بجانب الطريقة السابقة موضحة بالجدول الآتى بفرض أن a=6

| التخصيص التقليدى | الطريقة الحديثة | النتيجة |
|---|---|---|
| a=a+5 | a+=5 | 11 |
| a=a-5 | a-=5 | 1 |
| a=a*5 | a*=5 | 30 |

| التخصيص التقليدى | الطريقة الحديثة | النتيجة |
|---|---|---|
| a=a/3 | a/=3 | 2 |
| a=a+1 | a++ | 7 |
| a=a-1 | a-- | 5 |



**هناك فرق بين المؤثر = والمؤثر == حيث أن المؤثر = يستخدم كما سبق فى الحاق قيمة بمتغير أما المؤثر == يستخدم للمقارنة مثل if (a==b) معناها اذا كان a يساوى b**

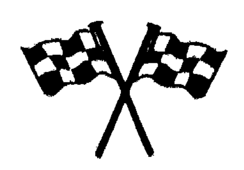
*مؤثر باقى خارج القسمة %*

**يستخدم لمعرفة باقى القسمة (و تستطيع أن تحدد هل الأرقام الموجودة فى متغير ما زوجية أم فردية) فمثلا اذا كانت A = 5**

**وكتبت : C = A % 2**

**فإن C ستكون قيمتها ١ وهو باقى قسمة الرقم ٥ على ٢**

**مؤثران الزيادة والنقصان Increment & Decrement من المعروف أن التعبير A = A +1 معناه زيادة قيمه المتغير A بمقدار واحد والتعبير A = A - 1 معناه إنقاص قيمه المتغير A بمقدار واحد ولكن توجد صورة آخرى مسموح بها لهتان العمليتان وهى A++ وتقابل A = A + 1 و A -- وتقابل A = A - 1**



# الفصل الثانى

# دوال الإدخال والاخراج

*فى هذا الفصل سوف نشرح دوال الإدخال والإخراج بأنواعها المختلفة والتى تتيح لك تطوير برامج أكثر فائدة وهى:*

- *دالة الطباعة على الشاشة* **printf ().**
- *دالة استقبال قيم من لوحة المفاتيح* **scanf ().**
- *دوال إدخال حرف واحد* **getchar(), getche()** *و* **getch()** *و دالة إظهار حرف واحد* **putchar()**
- *دالة طباعة و إدخال عبارة حرفية* **puts(),gets()**
- *دالة مسح الشاشة* **clrscr().**
- *دالة تغيير موضع المؤشر* **gotoxy ()**
- *دالة تغيرالألوان* **textcolor() , textbackground().**
- *دالة الطباعة والإدخال باستخدام الألوان* **cprintf() cscanf()**
- *دوال التعامل مع الطابعة.* **fprintf(), fputs(), fputc(), fwrite().**

شرحنا في الفصل الاول أن كل دالة مرتبطة بملف توجيه معين حيث يُستدعي هذا الملف في أول البرنامج بالعبارة #include فمثلا الدالة printf() معرفة بالملف stdioh وتكتب العبارة #include <stdio.h> في أول البرنامج حتى يتعرف المترجم على الدالة وهكذا مع باقي الدوال.في هذا الفصل سنشير إلى ملف التوجيه المعرّفة به كل دالة في بداية شرح الدالة.

## دالة الطباعة على الشاشة printf ( )

### ملف التوجيه : *stdio.h*

تستخدم الدالة printf() لطباعة البيانات بجميع أنواعها (int, char, string, float,...) على الشاشة فقط.

وتأخذ الدالة عدة صور وكذلك معاملات وأكواد تحدد شكل المخرجات.

وسنوضح فيما يلي مثال لكل صورة مع الشرح

*مثال :*

**printf ("Welcome with CompuScience")**

وفي هذا المثال يتم طباعة ما بين علامتي التنصيص " " كما هو على الشاشة وبالتالي نحصل على النتيجة التالية :

**Welcome with CompuScience**

*مثال*

**printf ("\n Welcome \nwith \n CompuScience ")**

وفي هذا المثال : الكود \n معناه new line أى سطر جديد وعندما يجد المترجم \n يترجمها الى سطر جديد وبالتالي نحصل على النتيجة التالية :

**Welcome**
**With**
**CompuScience**

وهناك أكواد أخرى تؤدى نتائج مختلفة فمثل الكود \t معناه tab أى مسافة جدولة خاليـة ويشمل الجدول رقم (١-٢) على الأكواد المستخدمة مع الدالـة printf() والتـى تـؤدى أشكال خرج مختلفة

| الكود | الإستخدام | مثال |
|---|---|---|
| \n | الإنتقال لسطر جديد | printf("\n") |
| \t | نقل المؤشر بعد ٨ مسافات | printf("\t") |
| \a | إخراج صوت الصافرة (بيب) | printf("\a") |
| \b | إرجاع المؤشر مسافة خلفية | printf("\b") |
| \xdd | طباعة الحرف المناظر للكود المكتوب بالنظام السادس عشر hexadecimal . | printf("\41")<br>النتيجة : A |
| \ddd | طباعة الحرف المناظر للكود المكتوب بالنظام الثمانى octal | printf("\101")<br>النتيجة : A |
| \\ | طباعة الشرطة المائلة | printf("\\") |
| \? | طباعة علامة الإستفهام | printf("\?") |
| \' | طباعة العلامة (') | printf("\'") |
| \" | طباعة علامة التنصيص | printf("\"") |

الجدول رقم (١-٢) الأكواد المستخدمة مع دالة **printf ( )**

و يوضح البرنامج الموجود فى شكل(١-٢) طريقة إستخدام هذه الأكواد

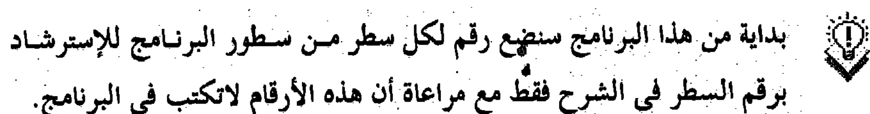
```
0:  /*Program Name cs2_1.c */
1:  #include <stdio.h>
```

```
2:  main( )
3:    {
4:        printf("\n this text display in new line");
5:        printf("\n word1\t tab1 \t tab2  \t  tab3.....");
6:        printf("\n Bell \a Bell \a Bell \b");
7:        printf("\n \" this line display Quotaions \"");
8:        printf("\n\xc9\xcd\xbb\n");
9:        printf("\xc8\xcd\xbc\n");
10:   }
```

شكل ١-٢ برنامج استخدام أكواد دالة printf()

وعن هذا البرنامج نوضح مايلى :





بداية من هذا البرنامج سنضع رقم لكل سطر من سطور البرنامج للإسترشاد برقم السطر في الشرح فقط مع مراعاة أن هذه الأرقام لاتكتب فى البرنامج.

- فى السطر رقم٤ الكود \n تعنى سطر جديد
- فى السطر رقم ٥ الكود \t تعنى مسافة tab أى مسافة جدولة
- فى السطر رقم ٦ الكود \a يعنى نغمة الصافرة (بيب) وهكذا
- وعند تنفيذ هذاالبرنامج ستحصل على النتيجة الموجودة بشكل ٢-٢

```
this text display in new line
word1  tab1     tab2     tab3 .......
Bell      Bell      Bell
"  this line display Quotaions  "
```



شكل ٢-٢ نتائج مختلفة لدالة printf()

## طباعة قيم المتغيرات على الشاشة

لطباعة القيم الموجودة بالمتغيرات نستخدم أكواد معينة لتحدد نوع البيانات المراد طباعتها بالدالة ()printf ، انظر المثال التالي :

```
1: printf("%d",a);
2: printf("%f",b);
```

في هذا المثال عندما يقابل مترجم اللغة العلامة ٪ ينظر الى الحرف التالي لهذه العلامه ويعتبر هذا الحرف توصيف لقيمة موجودة بعد العلامة , وكل حرف يحدد نوع معين من البيانات ففي هذا المثال نلاحظ :

| | | |
|---|---|---|
| %d | تعني | أي رقم صحيح int |
| %f | تعني | أي رقم حقيقي float |

وعلى ذلك فإن السطر رقم ١ معناه اطبع القيمة الصحيحة الموجودة بالمتغير a و السطر رقم ٢ معناه اطبع القيمة الحقيقية الموجودة بالمتغير b وهكذا

ويوضح جدول ٢-٢ أكواد طباعة أنواع البيانات الاخرى :

| الكود | الإستخدام | مثال |
|---|---|---|
| %d | توصيف لمتغير(أو ثابت) رقمي صحيح int | printf("%d",-10) |
| %f | توصيف لمتغير(أو ثابت) رقمي حقيقي float | printf("%f",5.7) |
| %c | توصيف لمتغير (أو ثابت) char (حرف واحد) | printf("%c","a") |
| %s | توصيف لعبارة حرفية string (حرف أو أكثر) | printf("%s","ab") |

| الكود | الإستخدام | مثال |
| --- | --- | --- |
| %u | توصيف لمتغير (أو ثابت) رقمى صحيح بدون إشارة | printf("%u",10) |
| %x | توصيف لمتغير (أو ثابت) بالنظام السادس عشر hex | printf("%x",af) |
| %o | توصيف لمتغير (أو ثابت) بالنظام الثمانى octal | printf("%o",67) |

جدول (٢-٢) أكواد طباعة البيانات

يشتمل البرنامج الموجود فى شكل ٣-٢ على مثال جامع يوضح معظم الأكواد المستخدمة ، وعند تنفيذ البرنامج ستحصل على النتيجة الموضحة بشكل ٤-٢ .

```
0:  /*Program Name : cs2_3.c */
1:  #include <stdio.h>
2:  main ( )
3:    {
4:       int A=5,B=10,C;
5:       float F=45.5;
6:       char ch='Y';
7:       char name[10]="Ahmed";
8:       long t=32142134;
9:       A=5;
10:      B=10;
11:      C=A+B;
12:      printf("\n the int C=%d",C);
13:      printf("\n the float F=%f",F);
14:      printf("\n the char ch=%c",ch);
15:      printf("\n the string name=%s",name);
```

```
16:     printf("\n the long no t=%ld",t);
17 :  }
```

شكل ٣-٢ مثال لاستخدام أكواد الطباعة

```
C = 15
F = 45.5
ch = Y
Name = Ahmed
t = 32142134
```

(شكل ٤-٢) نتيجة تنفيذ برنامج أكواد الطباعة

وعن هذا البرنامج نوضح مايلى :

- فى هذا البرنامج يشتمل السطر رقم ١ على الجملة #include<stdio.h> وتستخدم لتحميل ملف التوجيه stdio.h (لاضافة سطور الملف stdio.h الى ملف البرنامج CS2_2.C) الذى يحتوى على تعريف الدالة ()printf وفى السطر رقم ٢ الدالة الرئيسية ()main وفى السطر رقم ٣ تبدأ الدالة الرئيسية بالقوس } ثم فى السطر رقم ٤ اعلان عن ثلاث متغيرات من نوع صحيح int وفى السطر رقم ٥ اعلان عن المتغير F من نوع حقيقى( float)واعطائه قيمة ابتدائية 45.5 .
- فى السطر رقم ٦ اعلان عن المتغير ch من نوع حرف (char) واعطائه القيمة الابتدائية حرف Y
- فى السطر رقم ٧ اعلان عن المتغير name لتخزين عبارة حرفية واعطائه القيمة الابتدائية كلمة Ahmed
- السطر رقم ٨ اعلان عن المتغير t (صحيح طويل) واعطائه القيمة الابتدائية 32142134 ونلاحظ أن المتغير t أخذ النوع long لان القيمة التى خزنت فيه كبيرة.

- السطور رقم ٩ ، ١٠ ، ١١ لاعطاء قيم للمتغيرات A,B وقيمة C
- فى السطر رقم ١٢ الكود d%, يعنى اطبع قيمة المتغير الذى يلى العلامة ". وحرف d تعنى أنه صحيح وبالتالى يتم طباعة القيمة الموجودة بالمتغير C
- فى السطور رقم ١٣ ، ١٤ ، ١٥ ، ١٦ تطبع قيم المتغيرات F,ch,name,t و تنتهى الدالة الرئيسية بالقوس } وبالتالى ينتهى البرنامج

**ملاحظات**

يمكن أن تستخدم الأكواد f% و d% لتحديد عدد الارقام التى تظهر على الشاشة فمثلاً الصورة 3f.% يعنى طباعة ثلاث أرقام بعد العلامة العشرية فمثلا الرقم ٥٣٤,٥٦٣٧٨ يظهر بالصورة ٥٣٤,٥٦٤

لاحظ أننا شرحنا هذا البرنامج بالتفصيل لأنه أول برنامج ولكن فى الأمثلة التالية سوف نشرح المفاهيم أو الأوامر الجديدة فى البرنامج فقط.

## دالة الإدخال العامة scanf()

هى دالة الإدخال الرئيسية التى تسمح بإدخال جميع أنواع البيانات وهى تأخذ نفس المعاملات التى تأخذها الدالة printf() للتعامل مع البيانات و الموجودة بالجدول رقم (٢-٢).

***والصورة العامة للدالة scanf() هى***

```
int scanf (const char *format [, address, ...]);
```

والمثال الموجود بالشكل ٥-٢ يوضح استخدام الدالة scanf() حيث يقوم باستقبال مجموعة قيم مختلفة النوع وطباعتها على الشاشة

```
0:   /* Program Name : cs2_5c  */
1:   #include<stdio.h>
2:   main ()
```

```
3:    {
4:       int A,B,C;
5:       float R,S,T;
6:       char name[10];
7:       printf("\n\n Enter your name:");
8:       scanf("%s",name);
9:       printf("A=");
10:      scanf("%d",&A);
11:      printf("B=");
12:      scanf("%d",&B);
13:      printf("R=");
14:      scanf("%f",&R);
15:      printf("S=");
16:      scanf("%f",&S);
17:     printf("\n WELCOME    %s",name);
18:     printf("\n\n C=A+B=%d",A+B);
19:     printf("\n\n T=R+S=%f",R+S);
20:  }
```

شكل ٥-٢ استخدام الدالة ( ) scanf

وعن هذا البرنامج نوضح ما يلى:

- فى السطور رقم ٤ ، ٥ ، ٦ اعلان عن المتغيرات A,B,C ,R,S,T ,NAME
- فى السطر رقم ٧ تطبع الدالة()printf الرسالة :Enter your name
- فى السطر رقم ٨ تستقبل الدالة ()scanf العبارة الحرفية التى يدخلها المستخدم وتضعها فى المتغيرname.
- السطور من ٩ الى ١٦ تستخدم نفس فكرة السطرين ٧ و٨ فى إستقبال قيم المتغيرات
- وتلاحظ فى السطر رقم ١٠ أن الدالة ()scanf تستقبل قيمة صحيحة وتخزنها فى المتغير A ولكن ماذا يعنى المؤثر & ؟

&A تعنى تخزين القيمة الصحيحة فى المكان المخزن عنوانه فى المتغير A بمعنى أن A يشير الى عنوان المكان الذى تخزن فيه القيمة.وبالتالى العلامة & تجعل المتغير يشير الى عنوان المكان

```
Enter your name : Ahmed
A = 5
B = 10
R = 20
S = 30
Welcome Ahmed
C = A+b = 15
T = R + S = 50
```

شكل ٦-٢نتيجة إستخدام الدالة scanf() .

وفى هذا المستوى من الدراسة يكفى أن نعرف هذا القدر عن المؤثر & ولكن لنا مع هذا المؤثر & وقفة أخرى فى درس المؤشرات. مع ملاحظة أن المؤشر & يوضع فقط مع البيانات الصحيحة والحقيقية,(Int,float) والحرف ولا يوضع مع متغير العبارة الحرفية string.ولزيادة توضيح الدالة Scanf ()

انظر المثال التالى وتابع شرحه

```
scanf("%d,%f,%c",&a,&b,&ch);
```

هذا السطر معناه تخزين الرقم الصحيح d% فى أول متغير يلى الفاصلة وهو المتغير a ثم الرقم الحقيقى f% فى المتغير الذى يليه ، وكذلك الحرف فى المتغيرch مع ملاحظة أن الصورة d,%f,%c% تعنى أن المدخلات لابد ان تكون بنفس الصورة. أى بفاصل ، بين كل قيمة مدخلة.

وعند تنفيذ البرنامج ستحصل على النتيجة الموجودة بشكل ٦-٢

## دوال إدخال حرف واحد getchar(),getche(),getch()

بالرغم من أن الدالـة scanf() تسـتقبل جميـع أنـواع البيانـات الا ان لغـة C تشتمل على دوال اخرى تتعـامل مـع أنـواع خاصـة مـن البيانـات كـالحروف و العبـارات الحرفية ونوضح فيما يلى أهم هذه الدوال.

### الدالة getchar()

ملف التوجيه : *stdio.h*

تستخدم لإدخال حرف واحد ويظهر الحرف على الشاشة بعد الكتابة ولاتسـمح بالانتقال الى الامر التالى الا اذا ضغط المستخدم مفتاح الادخال (Enter) و هذا يشبه ما يحدث مع أمر Format فى نظام التشغيل DOS حيث يعطيك أمر Format الرسالة Format Another (y/n) وينتظر لتكتب حرف Y أو حرف N ويظهر الحرف وينتظر حتى تضغط على مفتاح الادخال Enter.

*مثال :*

```
char a;
a=getchar();
printf("%c",a);
```

### الدالة getche()

*ملف التوجيه* : conio.h

تستخدم لإدخال حرف واحد ويظهر هذا الحرف على الشاشة ولكنهـا تختلـف عن الدالة getchar() فى أنها لاتحتاج الى الضغط على مفتاح الادخال(Enter) للانتقال

للسطر التالي وتعمل هذه الدالة بطريقة مشابهة. لرسالة الخطأ التي تظهر في نظام التشغيل بهذا الشكل

(A)bort, (R)etry, (F)ail ?

حيث يتم التنفيذ بمجرد الضغط على أحد الحروف A,R,F بدون حاجة لضغط مفتاح الإدخال ويظهر الحرف على الشاشة

*مثال :*

```
char a;
a=getche();
printf("%c",a);
```

***الدالة getch ()***

***ملف التوجيه : conio.h***

تستخدم لإدخال حرف واحد ولكن تختلف عن الدالتين السابقتين في أن هذا الحرف لا يظهر على الشاشة وكذلك في أنها لاتحتاج الى الضغط على مفتاح الادخال Enterللانتقال للسطر التالي وهذا ما يحدث في كثير من الامثلة مثل حالة تنفيذ الامر dir/p في نظام التشغيل DOS حيث يتم عرض الفهرس صفحة بصفحة وينتظر الضغط على أى مفتاح للاستمرار وعند الضغط لا يظهر الحرف المضغوط عليه ولكن ينفذ الامر فقط.

*مثال :*

```
char a;
a=getch();
printf("%c",a);
```

## دالة طباعة حرف واحد putchar ()

ملف التوجيه : stdio.h

تستخدم لطباعة حرف واحد على الشاشة فمثلا الصورة: ;putchar('a') تطبع على الشاشة الحرف a كما هو

والمثال الموجود فى (شكل ٧-٢) يوضح كيفية استخدام دوال إدخال حرف وكذلك دالة طباعة حرف حيث يستقبل حرف باستخدام كل دالة لتوضيح الفرق بينهم.

```
0:    /* Program Name : cs2_7.c */
1-1: #include <stdio.h>
1-2: #include <conio.h>
2:    main ( )
3:      {
4:          char ch1,ch2,ch3;
5:          printf("\n ch1=");
6:          ch1=getchar();
7:          printf("\n ch2=");
8:          ch2=getche();
9:          printf("\nch3=");
10 :        ch3=getch();
11:         putch(ch1);
12:         putch(ch2);
13:         putch(ch3);
14:      }
```

شكل ٧-٢ استخدام دوال إدخال وطباعة حرف

وفي هذا البرنامج

- فى السطر رقم ٥ الدالة ( ) printf تطبع =ch1 بينما يشتمل السطر رقم ٦ على الدالة ()getchar وهى تستقبل حرف وتخزنه فى المتغير ch1
- وبالمثل نفهم السطور ٧و٨ والسطور ٩و١٠ مع ملاحظة الفرق فى طبيعة كل دالة
- ويشتمل السطر رقم ١٠ على الدالة ()getch التى تسمح باستقبال حرف من لوحة المفاتيح الا أن هذا الحرف لايظهر على الشاشة.
- وفى السطور رقم ١١و١٢و١٣ الدالة ()putch وتستخدم لطباعة الحروف المخزنة بالمتغيرات ch3,ch2,ch1

وعند تنفيذ البرنامج ستحصل على النتيجة التالية

```
ch1=a
ch2=b
ch3=
a
b
c
```

## دالة طباعة عبارة حرفية ()puts

*ملف التوجيه :* **stdio.h**

تستخدم لطباعة عبارة حرفية string حيث تطبع بدون توصيف شكل المخرجات

*مثال*

```
char name[10]="Ahmed"
puts(name);
puts("Mohamed");
```

وعند تنفيذ هذا المثال نحصل على النتيجة التالية

```
Ahmed
Mohmaed
```



الاعلان char name[10] معناه أن المتغير name من نـوع حرفى ويصلـح لتخزين كلمة اقصى عدد حروف لها هو ١٠ حروف

## دالة إدخال عبارة حرفية gets()

*ملف التوجيه :* **stdio.h**

وتستخدم الدالة gets() فى إدخال عبارة حرفية string انظر المثال التالى:

```
char name [20];
gets(name);
```

وفى هذا المثال تخزن الدالة gets() العبارة الحرفية فى المتغيرname

*مثال*

يوضح المثال الموجود بشكل ٨-٢ استخدام الدالتين gets () , puts ()

```
0:  /*Program Name : cs2_8.c*/
1:  #include <stdio.h>
2:  main ()
3:     {
4:      char name[10];
5:      printf("\n Enter your name:");
6:      gets(name);
7:      puts("welcome ");
8:      puts(name);
9:    }
```

شكل ٨-٢ استخدام دالتى gets () , puts ()

وعند التنفيذ ستحصل على النتيجة التالية :

```
Enter your name: Hesham
welcome
Hesham
```

وعن هذا البرنامج نوضح مايلى:

- فى السطر رقم ٥ دالة الطباعة ()printf تظهر الرسالة :Enter your name
- فى السطر رقم ٦ دالة إدخال العبارة الحرفية ()gets تخزن العبارة المدخلة فى المتغير name الذى تم الاعلان عنه فى السطر الرابع.
- فى السطر رقم ٧ الدالة ()puts تطبع الرسالة welcome
- فى السطر رقم ٨ الدالة ()puts تطبع محتويات المتغير name الذى أدخله المستخدم

مما سبق تجد ان لغة C تحتوى على دوال كثيرة للإدخال والإخراج مثل :

**printf(), scanf(), putch(), getchar(), getche(), getch(), puts(), gets()**

*والسؤال : لماذا تعددت ومتى نستخدم ايا منها؟*

تتعدد دوال الادخال والاخراج لتعطى المبرمج المرونة والقوة فى كتابة البرنامج حيث لديه أكثر من طريقة للتعامل مع البيانات وما تعجز عنه دالة تقوم به دالة أخرى وهكذا فمثلا الدالة ()scanf عند إستخدامها لإدخال عبارة حرفية فى متغير لاتسمح بوجود مسافات داخل نفس العبارة الحرفية حيث أنها تستخدم المسافة كفاصل بين المتغيرات المدخلة فى حين أن الدالة ()gets تستقبل البيانات الحرفية بشتى أنواعها بما فيها المسافة.

و أيضا كل دالة لها استخدام أمثل. فمثلا أذا أردت إدخال أو طباعة أكثر من قيمة فى سطر واحد وهذه القيم مختلفة النوع تكون أنسب دالة هى scanf(),printf()

أما أذا أردت إدخال أو طباعة عبارة حرفية string فتكون أنسب دالة هى gets() و puts() وإذا أردت إدخال أو طباعة حرف واحد تحدد طريقة الإدخال المطلوبة وتستعمل احدى الدوال getchar(),getche(),getch() و putch()

## دوال تحسين المدخلات والمخرجات

يحتاج المبرمج المحترف لإضافة بعض اللمسات الجمالية على شكل البيانات المدخلة والمخرجة لدفع الملل عن مستخدم البرنامج وسنشرح فيما يلى الدوال التى تساعدك على تحسين شكل المخرجات

### *دالة مسح الشاشة clrscr()*

ملف التوجيه: *conio.h*

تستخدم لمسح الشاشة ووضع المؤشر فى أول عمود من الصف الأول على الشاشة وتستخدم بالشكل التالى

```
clrscr()
```

### *دالة تغيير موضع المؤشر gotoxy()*

*ملف التوجيه:* *conio.h*

تستخدم لوضع المؤشر فى العمود x من الصف y وتأخذ الصورة التالية :

```
gotoxy(x,y)
```

*مثال :*

للانتقال بالمؤشر الى العمود رقم ٣٠ من الصف العاشر نستخدم الدالة بالصورة التالية :

```
gotoxy(30,10)
```

وهكذا تساعد هذه الدالة مع الدالة printf() على الكتابة فى اى مكان على الشاشة مما يساعد على تنسيق شكل المخرجات

### *دالة تغيير لون الكتابة textcolor()*

ملف التوجيه: *conio.h*

تستخدم لتغيير لون الكتابة التى ستطبع بعد الدالة وتأخذ الصيغة :

```
textcolor(color no)
```

أو الصيغة :

```
textcolor(color name)
```

حيث يتم تحديد اللون إما برقم اللون أو بإسمه ولابد من كتابة إسم اللون بالحروف الكبيرة فقط فمثلا للطباعة باللون الأزرق اكتب الدالة بالصورة الآتية :

```
textcolor(1)
```

أو

```
textcolor(BLUE)
```

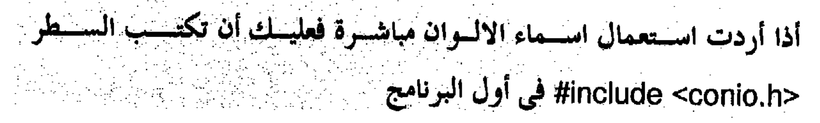
والجدول التالى ٣-٢ يوضح أكواد الألوان وأسمائها.

| اللون | رقم اللون | إسم اللون |
|---|---|---|
| اسود | ٠ | BLACK |
| أزرق | ١ | BLUE |
| أخضر | ٢ | GREEN |

| اللون | رقم اللون | إسم اللون |
| --- | --- | --- |
| سماوى | ٣ | CYAN |
| أحمر | ٤ | RED |
| بنفسجى | ٥ | MAGENTA |
| بنى | ٦ | BROWN |
| رمادى فاتح | ٧ | LIGHTGRAY |
| رمادى غامق | ٨ | DARKGRAY |
| أزرق فاتح | ٩ | LIGHTBLUE |
| أخضر فاتح | ١٠ | LIGHTGREEN |
| سماوى فاتح | ١١ | LIGHTCYAN |
| أحمر فاتح | ١٢ | LIGHTRED |
| بنفسجى فاتح | ١٣ | LIGHTMAGENTA |
| أصفر | ١٤ | YELLOW |
| أبيض | ١٥ | WHITE |

جدول ٣-٢ أكواد الألوان وأسمائها





أذا أردت استعمال اسماء الالوان مباشرة فعليك أن تكتب السطر #include <conio.h> فى أول البرنامج

يشتمل شكل ٩-٢ على برنامج بسيط يطبع الألوان التى يمكن استعمالها باستخدام الأرقام الدالة عليها

```
0:    /* Program Name : cs2_9.c */
1-1: # include <stdio.h>
1-2: #include <conio.h>
```

```
2:   int main(void)
3:     {
4:       int i;
5:       for (i=0; i<15; i++)
6:         {
7:           textcolor(i);
8:           cprintf("       Foreground Color=%d\r\n",i);
9:         }
10:      getch();
11:      return 0;
12:    }
```

شكل ٩-٢ برنامج طباعة الالوان

فى السطر رقم ٧ استخدمنا الدالة textcolor(i) لتغير لون الكتابة إعتمادا على قيمة التغير i الذى تتغير قيمته مع التكرار وقد إستخدمنا الدالة cprintf() بدلا من الدالة printf() حيث أن الأخيرة لا تعمل مع الألوان وعند استعمال الدالة cprintf() نستبدل العلامة \n بالعلامة \r\n

## دالة تغيير لون الخلفية textbackground()

ملف التوجيه : *conio.h*

وتستخدم لتغيير لون خلفية الكتابة التى ستطبع بعد تحديد لون الخلفية بها وتأخذ الصيغة التالية :

```
textbackground(color no);
```

أو الصيغة التالية :

```
textbackground(color name);
```

و معاملات هذه الدالة هى نفس معاملات الدالة السابقة textcolor() مع ملاحظة أن الدالة textbackground() لاتستخدم سوى الألوان من رقم ١ الى رقم ٧ المذكورين فى الجدول رقم ٣-٢

ويشتمل شكل ١٠-٢ على مثال يوضح استخدام الالوان مع الكتابة والخلفية

```
0:    /*Program Name :cs2_10.c */
1-1: #include <stdio.h>
1-2: #include <conio.h>
2:    int main(void)
3:        {
4:           int i, j;
5:           clrscr();
6:           for (i=0; i<9; i++)
7:              {
8:                 for (j=0; j<5; j++)
9:                     cprintf("Allah ");
10:                cprintf("\r\n");
11:                textcolor(i+1);
12:                textbackground(i);
13:             }
14:          return 0;
15:       }
```

شكل ١٠-٢ برنامج يوضح استخدام الألوان مع الكتابة والخلفية

وعن هذا البرنامج نوضح مايلى :

- فى السطر رقم ٩ الدالة ;("cprintf("Allah تطبع الكلمة باللون الحالى
- فى السطر رقم ١٠ الدالة ;("cprintf("\r\n لطباعة سطر جديد
- فى السطر رقم ١١ الدالة ;textcolor(i+1) تغير لون الكتابة بالون رقم i+1

- في السطر رقم ١٢ الدالة ;textbackground(i) تغير لون الخلفية باللون رقم i

## دوال الإدخال والإخراج التي تستخدم الألوان

سبق أن أشرنا إلى أن دالة الطابعة ()printf ودالة الإدخال ()scanf وكذلك باقي الدوال التي تم شرحها قد صممت بحيث تعمل باللون الابيض على الاسود ولا تتأثر بدوال تغيير الالوان فإن هناك مجموعة من الدوال المقابلة للدوال السابقة والتي صممت للتعامل بالالوان المحددة وكلها مسبوقة بالحرف c مثل

cprintf(),cscanf(),cputs(),cgets(),..................

وكلها معرفة في الملف conio.h



إذا إستخدمت الدالة ()textbackground لتحديد لون الخلفية وأتبعتها بالدالة ()clrscr سيتم مسح الشاشة باللون الذى تم تحديده ويصبح هذا اللون هو لون خلفية الشاشة كلها أما اذا لم تستخدم الدالة ()clrscr فان لون الخلفية سيخصص للكلمات التي تكتب فقط بينما يبقى لون الشاشة كما هو

نفذ البرنامج الموجود بشكل ١١-٢ ولاحظ الألوان التي ستحصل عليها

```
0:  /*Program Name : cs2_11 */
1:  #include <stdio.h>
2:  #include <conio.h>
3:  main ()
4:      {
5:        textbackground(BLUE);
6:        clrscr();
7:        textcolor(RED);
8:       cprintf("\r\n  THIS TEXT  DISPLAYED WITH RED ON
                                        BLUE COLOR");
9:       getch();
10:      textbackground(GREEN);
```

```
11:     clrscr();
12:     textcolor(YELLOW);
13:     cprintf("\r\n  THIS TEXT  DISPLAYED WITH YELLOW
                                          ON RED COLOR");
14:  }
```

شكل ١١-٢ برنامج لتغيير الألوان

وعن هذا البرنامج نوضح مايلى

- فى السطر رقم ٥ تغير الدالة ;textbackground(BLUE) لـون خلفيـة الكتابـة الى اللون الازرق
- فى السطر رقم ٦ تمسح الدالة ;()clrscr الشاشة (يصبح لونها أزرق)
- فى السطر رقم ٧ الدالة ;textcolor(RED) تغير لون الكتابة الى اللون الاحمر وهكذا

لاحظ r\n\ : تقابل n\ فى حالة الدالة ()printf

## دوال التعامل مع الطابعة

من الأمور المهمة للمبرمج المبتدىء والمحترف على السواء إخراج البيانات على الطابعة والتعامل مع الطابعة.

ولغة C تمكننا من التعامل مع الطابعة من حيث إخـراج البيانات وإختبار حالة الطابعة وهل هى تعمل أم مغلقة وهل الورق موجود بالطابعة أم لا وفى الفقرة التالية سوف نتعرض لأبسط صورة للتعامل مع الطابعة وهى الطباعة فقط أما باقى الحالات فسنتعرض لها فيما بعد

وتستخدم الدالة ()fprintf للطباعة على الطابعة ولكن مع معرف الطابعة القياسى stdprn ويظهر كأول معامل فى الدالة ()fprintf والمتغير stdprn يعنى الطابعة ومعناه

اطبع هذه الرسالة على الطابعة ومعنى ذلك أن هناك معرفات أخرى لها معنى مختلف سوف نتعرض لها لاحقا.

وتشبه الدالة fprintf() الدالة printf() حيث تستخدم نفس المعاملات التي تستخدمها الا أن المخرجات تظهر على الطابعة. وكذلك باقي صور الدالة printf() مسموح بها مع fprintf()

جرب البرنامج التالي :

```
0:  /* Program Name :cs2_12.c
1:  #include <stdio.h>
2:  main ()
3:    {
4:      fprintf(stdprn," These words go to the printer............");
5:    }
```

شكل ٢-١٢ برنامج للطباعة على الطابعة

ستحصل على الرسالة التالية :

These words go to the printer على الطابعة.

# تذكر

| عندما تريد | إستخدم الدالة |
|---|---|
| طباعة حروف أو أرقام | printf() |
| طباعة عبارات حرفية فقط | puts() |

| عندما تريد | إستخدم الدالة |
|---|---|
| إدخال حروف أو أرقام | scanf() |
| إدخال عبارات حرفية | gets() |
| إدخال حرف دون إظهاره على الشاشة | getch() |
| إدخال حرف مع إظهاره على الشاشة | getche() |
| مسح الشاشة | clrscr() |
| تغيير موضع المؤشر | gotoxy() |
| تغيير ألوان الكتابة | textcolor() |
| تغيير ألوان خلفية الكتابة | textbackground() |
| الطباعة بالألوان على الشاشة | cprinf(),cputs |
| إدخال البيانات بالألوان | cscanf(),cgets |
| الطباعة على الطابعة | fprintf() |



# الفصل الثالث

# الدوارات LOOPS

*فـي هـذا الفصـل سنشـرح موضوعـا مـن الموضوعات المهمة وهو موضـوع الـداورات ونقصـد بـه تكـرار تنفيـذ مجموعـة مـن الأوامـر أكـثر مـن مـرة. وذلـك مـن خـلال الموضوعات التالية*

- *الدوارة* **for**
- *تنفيذ أكثر من جملة مع الدوارة* **for**
- *الدوارات المتداخلة بإستخدام* **for**
- *الدوارة اللانهائية بإستخدام* **for**
- *الدوارة* **while**
- *الفرق بين* **For** *و* **while**
- *الدوارة اللانهائية بإستخدام* **while**
- *الدوارة* **do ........while**
- *الفرق بين* **while** *و* **do.....while**
- *الدوارة اللانهائية بإستخدام* **do......while**

## الدوارة for

تستخدم لتكرار تنفيذ عملية عدد محدد من المرات وتأخذ الصيغة العامة التالية :

```
for(inital-value;condition;increment)
    statments;
```

**حيث**

Inital-Value هي القيمة الابتدائية

Condition هو شرط انهاء التكرار

increment هي قيمة الزيادة الدورية

ومعناها ابدأ العد من القيمه الابتدائيه (inital-Value) طالما أن الشرط (Condation) صحيح ومقدار الزيادة كل مرة هو increment.

*مثال ١*

البرنامج الموجود في شكل ١-٣ يستخدم الدوارة for لطباعة كلمة Allah عشر مرات كل كلمة في سطر مستقل

```
0: /* Program Name :cs3_1.c */
1: #include <stdio.h>
2:  main()
3:  {
4:     int i;
5:       for(i=0;i<10;i++)
6:            printf("\n Allah");
7:   }
```

شكل ١-٣ استخدام الدوارة for

وعن هذا البرنامج نوضح مايلى :

- فى السطر رقم ٤ اعلان عن متغير صحيح من نوع int
- فى السطر رقم ٥ جملة for حيث يبدأ بقيمة للمتغير i تساوى صفر ثم يزيدها كل مرة واحد حتى تصل القيمة الى ١٠ وفى كل مرة يطبع كلمة Allah فى سطر مستقل
- فى السطر رقم ٦ الجملة printf("\n allah"); يتم تنفيذها ١٠ مرات وكل مرة يتم زيادة قيمة المتغير i واختبار قيمته هل وصلت الى القيمة ١٠ أم لا فاذا وصلت القيمة الى ١٠ يتوقف التكرار

وعند تنفيذ هذا البرنامج ستحصل على النتيجة التالية :

```
Allah
Allah
Allah
---
---
Allah
```

*مثال ٢*

والبرنامج الموجود فى شكل ٢-٣ يطبع الارقام من صفر إلى ٩

```
0:  /*Program Name : cs3_2.c */
1:  #include <stdio.h>
2:  main()
3:    {
4:       int i
5:       for(i=0;i<10;i++)
6:         printf("\n i=%d",i);
7:    }
```

شكل ٢-٣ برنامج طباعة الأرقام من صفر إلى ٩

## مثال ٣

والبرنامج الموجود فى شكل ٣-٣ يطبع شفرة Ascii وهى الاكـواد المقابلـة للأحـراف القابلة للطباعة وغير القابلة للطباعة

```
0: /* Program Name : cs3_3.c   */
1: /*Ascii example */
2: #include <stdio.h>
3: #include <conio.h>
4: main ()
5: {
6:   int i;
7:    for (i=0;i<256;i++)
8:        printf ("%d=%c\t",i,i);
9: }
```

شكل ٣-٣ برنامج طباعة شفرة Ascii

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 000 | | (nul) | 016 | ► | (dle) | 032 | sp | 048 | 0 | 064 | @ | 080 | P | 096 | ` | 112 | p |
| 001 | ☺ | (soh) | 017 | ◄ | (dc1) | 033 | ! | 049 | 1 | 065 | A | 081 | Q | 097 | a | 113 | q |
| 002 | ☻ | (stx) | 018 | ↕ | (dc2) | 034 | " | 050 | 2 | 066 | B | 082 | R | 098 | b | 114 | r |
| 003 | ♥ | (etx) | 019 | ‼ | (dc3) | 035 | # | 051 | 3 | 067 | C | 083 | S | 099 | c | 115 | s |
| 004 | ♦ | (eot) | 020 | ¶ | (dc4) | 036 | $ | 052 | 4 | 068 | D | 084 | T | 100 | d | 116 | t |
| 005 | ♣ | (enq) | 021 | § | (nak) | 037 | % | 053 | 5 | 069 | E | 085 | U | 101 | e | 117 | u |
| 006 | ♠ | (ack) | 022 | ▬ | (syn) | 038 | & | 054 | 6 | 070 | F | 086 | V | 102 | f | 118 | v |
| 007 | • | (bel) | 023 | ↨ | (etb) | 039 | ' | 055 | 7 | 071 | G | 087 | W | 103 | g | 119 | w |
| 008 | ◘ | (bs) | 024 | ↑ | (can) | 040 | ( | 056 | 8 | 072 | H | 088 | X | 104 | h | 120 | x |
| 009 | | (tab) | 025 | ↓ | (em) | 041 | ) | 057 | 9 | 073 | I | 089 | Y | 105 | i | 121 | y |
| 010 | | (lf) | 026 | | (eof) | 042 | * | 058 | : | 074 | J | 090 | Z | 106 | j | 122 | z |
| 011 | ♂ | (vt) | 027 | ← | (esc) | 043 | + | 059 | ; | 075 | K | 091 | [ | 107 | k | 123 | { |
| 012 | ♀ | (np) | 028 | ∟ | (fs) | 044 | , | 060 | < | 076 | L | 092 | \ | 108 | l | 124 | \| |
| 013 | | (cr) | 029 | ↔ | (gs) | 045 | - | 061 | = | 077 | M | 093 | ] | 109 | m | 125 | } |
| 014 | ♫ | (so) | 030 | ▲ | (rs) | 046 | . | 062 | > | 078 | N | 094 | ^ | 110 | n | 126 | ~ |
| 015 | ☼ | (si) | 031 | ▼ | (us) | 047 | / | 063 | ? | 079 | O | 095 | _ | 111 | o | 127 | ⌂ |

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 128 | Ç | 143 | Å | 158 | ₧ | 172 | ¼ | 186 | ║ | 200 | ╚ | 214 | ╓ | 228 | Σ | 242 | ≥ |
| 129 | ü | 144 | É | 159 | ƒ | 173 | ¡ | 187 | ╗ | 201 | ╔ | 215 | ╫ | 229 | σ | 243 | ≤ |
| 130 | é | 145 | æ | 160 | á | 174 | « | 188 | ╝ | 202 | ╩ | 216 | ╪ | 230 | µ | 244 | ⌠ |
| 131 | â | 146 | Æ | 161 | í | 175 | » | 189 | ╜ | 203 | ╦ | 217 | ┘ | 231 | τ | 245 | ⌡ |
| 132 | ä | 147 | ô | 162 | ó | 176 | ░ | 190 | ╛ | 204 | ╠ | 218 | ┌ | 232 | Φ | 246 | ÷ |
| 133 | à | 148 | ö | 163 | ú | 177 | ▒ | 191 | ┐ | 205 | ═ | 219 | █ | 233 | Θ | 247 | ≈ |
| 134 | å | 149 | ò | 164 | ñ | 178 | ▓ | 192 | └ | 206 | ╬ | 220 | ▄ | 234 | Ω | 248 | ° |
| 135 | ç | 150 | û | 165 | Ñ | 179 | │ | 193 | ┴ | 207 | ╧ | 221 | ▌ | 235 | δ | 249 | ∙ |
| 136 | ê | 151 | ù | 166 | ª | 180 | ┤ | 194 | ┬ | 208 | ╨ | 222 | ▐ | 236 | ∞ | 250 | · |
| 137 | ë | 152 | ÿ | 167 | º | 181 | ╡ | 195 | ├ | 209 | ╤ | 223 | ▀ | 237 | φ | 251 | √ |
| 138 | è | 153 | Ö | 168 | ¿ | 182 | ╢ | 196 | ─ | 210 | ╥ | 224 | α | 238 | ε | 252 | ⁿ |
| 139 | ï | 154 | Ü | 169 | ⌐ | 183 | ╖ | 197 | ┼ | 211 | ╙ | 225 | ß | 239 | ∩ | 253 | ² |
| 140 | î | 155 | ¢ | 170 | ¬ | 184 | ╕ | 198 | ╞ | 212 | ╘ | 226 | Γ | 240 | ≡ | 254 | ■ |
| 141 | ì | 156 | £ | 171 | ½ | 185 | ╣ | 199 | ╟ | 213 | ╒ | 227 | π | 241 | ± | 255 | |
| 142 | Ä | 157 | ¥ | | | | | | | | | | | | | | |

شفرة Ascii Code

وفى البرنامج الموجود فى الشكل رقم ٣-٣ فى السطر رقم ٧ تقوم الدوارة for بتغير قيمة المتغير من القيمة ٠ الى القيم ٢٥٥ وكل مرة تقوم الدالة ()printf بطباعة قيمة المتغير i والحرف المقابل لهذا الكود وبالتالى نحصل على جدول شفرة Ascci

## تنفيذ أكثر من جملة مع for

فى جملة for السابقة كنا نطلب تكرار تنفيذ جملة واحدة عدد محدد من المرات لكن ماهو العمل اذا كان المطلوب تكرار تنفيذ أكثر من جملة لعدد محدد من المرات. بعبارة اخرى تكرار تنفيذ بلوك كامل داخل البرنامج أكثر من مرة هذا ماسنحاول الاجابة عليه فيما يلى :

لو أردنا تنفيذ أكثر من جملة لعدد محدد من المرات يجب وضع القوس } فى بداية البلوك المراد تكراره ووضع القوس { فى نهاية البلوك.

*مثال*

أقرا المثال الموجود بشكل ٤-٣ واكتبه وتفحصه جيدا وقبل أن تنفذه حاول الاجابة على السؤال التالى :

ماذا تعتقد أن يكون شكل المخرجات ؟

```
0:/*Program Name : cs3_4.c */
1: #include <stdio.h>
2:  main()
3:  {
4:     int i;
5:     for(i=0;i<10;i++)
6:         printf("\n wlecome ");
7:     printf("with     ");
```

```
8:    printf("CompuScience");
9:  }
```

شكل ٤-٣استخدام الدوارة for

من أول نظرة للبرنامج الموجود بالشكل ٤-٣ قد تظن أن المخرجات طباعة الكلمات الاتية ١٠ مرات

**Welcome With CompuScience**

ولكن لونفذنا البرنامج سيكون الناتج هو طباعة الكلمة Welcome ١٠ مرات ثم طباعة الكلمات With CompuScience مرة واحدة هل عرفت ماهو السبب ؟؟؟

السبب هو أن جملة for تأخذ أول أمر يليها فقط وتنفذه ثم بعد ذلك تنتقل الى الأوامر التالية.

ولكن كيف يمكننا طباعة الثلاث جمل ١٠ مرات ؟

لابد من تحديد أول هذه الجمل بالقوس } وفي نهاية الجمل القوس{ فيصبح البرنامج السابق كما في الشكل ٥-٣

```
/* Program Name :cs3_5.c  */
/* program for print 10 once   */
1: #include <stdio.h>
2:  main()
3:  {
4:    int i;
5:      for(i=0;i<10;i++)
6:          {
7:              printf("\n wlecome ");
8:              printf("with  ");
9:              printf("CompuScience");
```

```
10:          }
11:   }
```

شكل ٥-٣ برنامج تكرار تنفيذ مجموعة جمل

وبالتالى عند تنفيذ البرنامج الموجود بشكل ٥-٣ ستحصل على عبارة

**Welcome With CompuScience**

عدد ١٠ مرات وذلك نتيجة اضافة السطر رقم ٦ أى القوس } قبل الجمل المراد تكرارها والسطر رقم ١٠ أى القوس { فى نهاية الجمل المراد تكرارها

*تغير معدل الزيادة*

فى الأمثلة السابقة إستخدمنا تعبير الزيادة ++i ومعناه زيادة قيمة المتغير بمقدار واحدفى كل مرة

ولكن فى بعض الأحيان نحتاج أن يكون مقدار الزيادة أكثر أو أقل من واحد حينئذ يلزم تغيير هذاالتعبير ++i الى الصورة المطلوبة.

مثلا 2 = + i معناها i = i + 2 أى زيادة قيمة i كل مرة بمقدار إثنين

0.5 = + i معناها i=i+0.5 أى زيادة قيمة i كل مرة بمقدار 0.5 وهكذا

والبرنامج الموجود فى شكل ٦-٣ يستعمل الدوارة for ولكن بمعدل زيادة مقداره ٢ فى كل مرة

```
0:/* Program Name : cs3_6.c  */
1: #include <stdio.h>
2:  main()
3:  {
4:     int i;
5:       for(i=0;i<10;i+=2)
```

```
6:         printf("\n i=%d",i);
7:  }
```

شكل ٦-٣ طباعة الأرقام بمعدل زيادة ٢

وعند تنفيذ البرنامج ستحصل على النتيجة التالية :

```
i=0
i=2
i=4
.....
i=8
```

### *تغيير معدل الزيادة بالسالب*

فى بعض الاحيان نحتاج أن يكون معدل التغيير بالسالب كما فى بعض العمليات الحسابية مثلا ١٠ ، ٩ ، ٨ .... وهكذا

والبرنامج الموجود وبشكل ٧-٣ يقوم بطباعة الارقام من ١٠ الى ١ وذلك باستعمال خطوة سالبة مع الدوارة for

```
0:/* Program Name : cs3_7.c  */
1: #include <stdio.h>
2:  main()
3:  {
4:     int i;
5:     for(i=10;i>0;i--)
6:         printf("\n i=%d",i);
7:  }
```

شكل ٧-٣ طباعة الارقام بخطوة سالبة

وعند تنفيذ البرنامج نحصل على النتيجة التالية :

```
i=10
```

```
i=9
i=8
....
i=1
```

وفى هذا البرنـامج يشـتمل السـطر رقـم ٥ علـى الجملـة (--for(i=10;i>0;i و معناها أبدأ بوضع القيمة ١٠ فى المتغير i ثم انقص هذه القيمة بمقدار واحد كـل مـرة نتيجة للتعبير --i والشرط أن i أكبر من القيمة صفر وبالتالى يتم العد التنازلى.

## الدوارات المتداخلة بإستخدام for

الدورات المتداخلة عبارة عن دوارة كبيرة تشتمل بداخلها على دوارة أو أكثر. بمعنى أن مجموعة التعليمات الموجودة بالدوارة الداخلية يتكـرر تنفيذهـا طالمـا لـم ينتـه العداد فاذا انتهى ينتقل التنفيـذ إلـى الـدوارة الخارجيـة ويتكـرر تنفيـذ تعليمـات الـدوارة الخارجية حتى ينتهى العداد المحدد لها . وتشبه فكـرة الـدورات المتداخلـة فكـرة عمـل عقارب الساعه فتجد عقرب الثوانى يدور ٦٠ دورة فيدور عقرب الدقـائق بمقـدار دقيقـه وهكذا.

والبرنامج الموجود فى الشكل رقم ٨-٣ يسـتخدم فكـرة الـدورات المتداخلـة لطباعة الأرقام من صفر إلى ٩ باستخدام الدوارة الداخلية ثم يطبع الأرقام من صفر إلى ٥ باستخدام الدوارة الخارجيـة. بحيـث نحصـل فـى النهايـة علـى طباعـة رقـم مـن الـدوارة الخارجية مقابل طباعة عشرة أرقام من الدوارة الداخلية.

```
0:/* Program Name : cs3_8.c  */
1: #include <stdio.h>
2:  main()
3:  {
4:    int i,j;
5:      for(i=0;i<5;i++)
6:        {
```

```
7:        for(j=0;j<10;j++)
8:          {
9:            printf("\n i=%d",i);
10:             printf("\t j=%d",j);
11:           }       /*  first for loop    */
12:       }          /*  secound for loop */
13:  }               /*  main() function   */
```

شكل ٨-٣ استخدام الدوارات المتداخلة

ومن الامثلة المفيدة لاستخدام الدورات المتداخلة استخدام الدوارة for لطباعة جدول الضرب من أول ١×١ الى ١٢ × ١٢ حيث نستعمل الدوارة الخارجية لتغيير الرقم المضروب ونستعمل الدوارة الداخلية لتغير الرقم المضروب فيه من ١ الى ١٢ كما يتضح من البرنامج الموجود فى الشكل رقم ٩-٣

```
0:/* Program Name :cs3_9.c  */
1: #include <stdio.h>
2: main()
3:  {
4:    int i,j;
5:     for(i=1;i<13;i++)
6:        {
7:          printf("\n");
7:          for(j=i;j<13;j++)
8:            {
9:              printf(" IxJ=%d",i*j);
10:
11:           }    /*  first for        */
12:       }        /*  secand for       */
13:   }            /*  main() function  */
```

شكل ٩-٣ برنامج استخدام الدوارات المتداخلة لطباعة جدول الضرب

أكتب البرنامج ونفذه وتأكد من النتيجة

ولتحسين البرنامج بحيث لا يطبع القيم المكررة مثل ١×٢ و ٢×١ نقوم بتغير بداية الدوارة الثانية بحيث تبدأمن القيمة i. فيصبح السطر رقم ٧ كما يلى:

```
for (j = i;j <13;j++)
```

## الدوارة اللانهائية بإستخدام for

ومعناها تكرار تنفيذ الجملة بدون شرط ولايتوقف التنفيذ حتى يضغط المستخدم Ctrl+c او Ctrl+break وتأخذ الدوارة اللانهائية باستخدام for الصورة

```
for(;;)
```

والمثال التالى يستعمل الدوارة اللانهائية للاستمرار فى طباعة كلمة Allah مالم يضغط المستخدم على Ctrl+break فاذا ضغط المستخدم على Ctr+c تنتهى الدوارة ويتوقف التنفيذ

```
#include <stdio.h>
  main ()
    {
      for(;;)
        printf("\t Allah");
    }
```



لوكتبت الدوارة بالصورة for(;;) وأتبعتها بالعلامة ; وحاولت تنفيذ الملف النهائى (EXE.) فان الجهاز يدخل فى مسار لانهائى(hang) ولاتستطيع حل هذه المشكلة الا بغلق الجهاز وتشغيله مرة أخرى ويمكنك استخدام هذه الفكرة مع برنامج كلمة السر فإذا أدخل المستخدم كلمة سر خطأ لا يستجيب الجهاز لاى أمر ولابد من غلقه.

## الدوارة while

تستخدم الدوارة while لتكرار تنفيذ جملة أو مجموعة جمل عدد من المرات غير معلوم العدد وانما يتوقف هذا العدد على شرط موجود بالدوارة وتأخذ الدوارة while الصورة التالية :

```
while(condition)
  statment;
```

ومعناها طالما أن الشرط (condition) صحيح نفذ الجملة (Statement)

وهى تقوم بتكرار الجملة أو مجموعة الجمل التابعة لها طالما كان شرط التكرار صحيح وعندما يصبح شرط التكرار غير صحيح ينوقف تنفيذ الدوارة

والبرنامج الموجود فى الشكل ١٠-٣ يستخدم الدوارة while لطباعة كلمة Allah ١٠ مرات كل مرة فى سطر مستقل وهذا الاستخدام للدوارة while غير جيد ولكننا نكتب هذا البرنامج لتوضيح أجزاء الدوارة while

```
0: /* Program Name : cs3_10.c  */
1:#include <stdio.h>
2:  main ()
3:    {
4:      int i=0;
5:      while(i<10)
6:        {
7:          printf("\n Aallah");
8:          i++;
9:        }
10:   }
```

شكل ١٠-٣ استخدام while

وعن هذا البرنامج

- فى السطر رقم ٤ إعلان عن متغير من نوع صحيح هو i
- السطر رقم ٥ معناه طالما أن قيمة i أقل من ١٠ نفذ الجمل التالية وبالفعل أول مرة ستكون قيمة لمتغير i تساوى صفر وبالتالى يتم تكرار الدوارة
- السطر رقم ٧ معناه نفذ الجملة ;printf("\n Allah") طالما أن الشرط صحيح وهو طباعة كلمة Allah
- السطر رقم ٨ زيادة قيمة المتغير i بمقدار ١ ثم يعود البرنامج الى اختبار شرط الدوارة while فاذا كانت قيمة i ماتزال أقل من ١٠ يتكرر تنفيذ الجمل التالية وهكذا حتى يصبح الشرط غير صحيح

**ملاحظات**

١. لاحظ أن كتبنا القوس } بعد while لاننا نرغب فى تكرار التنفيذ على أكثر من جملة (بلوك) والقوس { فى سطر مستقل لاغلاق الدوارة
٢. لتغير قيمة الزيادة أو جعلها بالسالب غير التعبير ++i كما نشاء كما تم مع الدورة for

## الفرق بين for و while

الدوارة for دوارة عددية حيث تعتمد على العداد وينتهى التكرار فيها بانتهاء عدد مرات التكرار أما الدوارة while فدوارة شرطية أى تعتمد على الشرط الذى يلى الأمر while حيث تتكررالجمل التى تليها طالما كان الشرط صحيحا وتنتهى الدوارة بكسر هذا الشرط. وبالتالى الاستخدام الامثل للدوارة for هو تكرار عملية أكثر من مرة بشرط أن يكون عدد مرات التكرار معلوم والاستعمال الامثل للدوارة while هو التكرار بناء على شرط معين

*مثال*

يقوم البرنامج الموجود بشكل ١١-٣ باستقبال حروف من لوحة المفاتيح ويستمر فى ذلك حتى تضغط مفتاح Enter وعند ضغط مفتاح Enter يتوقف قبول الحروف ويطبع البرنامج عدد الحروف التى أدخلتها.

```
1/* Program Name : cs3_11.c  */
2:  #include <stdio.h>
3:  #include <conio.h>
4:  main ()
5:    {
6:        int count=0;
7:        printf ("\n ENTER CHARCTERS..\n");
8:        while (getche() !='\r')
9:            count++;
10:       printf ("\n charcter count is %d ",count);
11:   }
```

شكل ١١-٣ برنامج عد الحروف المدخلة

وعن البرنامج الموجود بالشكل ١١-٣ نوضح مايلى :

فى السطر رقم ٨ تقوم الدالة getche() باستقبال حرف وتقوم الدوارة while باختبار هذا الحرف فاذا كان هذا الحرف هومفتاح الادخال Enter يتوقف عمل الدوارة واذا كان أى حرف آخر يقوم السطررقم ٩ بزيادة قيمة المتغير count وهكذا حتى يضغط المستخدم على مفتاح الادخال وفى النهاية يطبع البرنامج عدد الحروف المدخلة.

## الدوارة اللانهائية بإستخدام while

سبق أن قلنا أن الدوارة اللانهائية مع for تـأخذ الصـورة (;;) for أما الـدوارة اللانهائية مع while فتـأخذ الصـورة (while(1 وهـى أكـثر اسـتعمالا مـن الـدوارة for والبرنامج الموجود بشكل ١٢-٣ يطبع قيم متواليـة ٠و١و٢ وهكـذا وذلـك باسـتعمال دوارة لانهائية حتى يضغط المستخدم على Ctrl+C.

```
/* Program Name : cs3_12.c */
 #include <stdio.h>
  main ()
   {
      int i=0;
      while(1)
        {
          printf ("i=%d",i);
           i++;
        }
   }
```

شكل ١٢-٣ الدورة اللانهائية

## الدوارة do ........ while

تستخدم الدوارة do.... while لتكرار تنفيذ جملة أو مجموعة جمـل أكـثر مـن مرة بناء على شرط معين كما هو الحال مع الدوارة while ولكن الفرق بينهما أن الدوارة while تختبر الشرط أولا فاذا كان صحيحا تنفذ الجمل التالية لها والا فلا ، أمـا الـدوارة do.....while فتنفذ الجمل التالية لهـا أولا ثـم تختـبر الشـرط. فـاذا كـان صحيحـا تعيـد التنفيذ والا توقف التكرار.

وتأخذ الدوارة while ..... do الصيغة

```
do
  {
    statement1;
  }while (condition);
```

ومعناها do أي نفـذ الجمـل التاليـه وهـى Statement1 ومـا يليهـا طالمـا كـان الشـرط (Cndertion) صحيح.

والبرنامج الموجود بشكل ١٣-٣ يستخدام الدوارة do ...while لطباعة رسالة يطلب فيها من المستخدم ادخال كلمة سر ومقارنتها بالكلمة المخزنة فاذا كانت الكلمـة المدخلة صحيحة انتهى البرنامج والا يتم تكرار استقبال كلمة السر مرة اخرى.

```
1:  /* Program name : cs3_13.c  */
2:  /* password...*/
3:  #include <stdio.h>
4:  #include <conio.h>
5:    main ()
6:      {
7:          char pass[10];
8:          do
9:            {
10:             printf ("\n Enter Password:");
11:             scanf ("%s",pass);
12:            } while (strcmp(pass, "azab") !=0);
13:     }
```

شكل ١٣-٣ برنامج كلمة السر

وعن هذا البرنامج نوضح مايلى:

- فى السطر رقم ٨ بداية الدوارة do والسطر رقم ١٠ يطبع الرسالة Enter Password: على الشاشة
- والسطر رقم ١١ يستقبل كلمة ويخزنها فى المتغير pass والسطر رقم ١٣ يقارن بين الكلمة التى أدخلت وكلمة azab فاذا كانتا متطابقتين ينتهى عمل do..while والا عاد البرنامج الى بداية do.
- نفذ البرنامج وأدخل كلمة سر غير الواردة بالبرنامج وهى azab مرة واثنيين قبل كتابة كلمة السر الصحيحة ولاحظ تنفيذ البرنامج.



الدالة () strcmp تقوم بمقارنه متغيرين من نوع عبارة حرفيه string فإذا كان المتغيرين متطابقين كان الفرق بينهما صفر.

*مثال*

البرنامج الموجود بالشكل ١٤-٣ تعديل لبرنامج كلمة السر السابق بحيث لا تظهر كلمة السر التى يكتبها المستخدم وهى تكتب على الشاشة حتى لايراها شخص آخر وتعتمد فكرته على تغير الالوان.

اكتب البرنامج ونفذه ولاحظ ذلك.

```
1:  /* Program Name : cs3_14.c  */
2:  /* password...*/
3:    #include <stdio.h>
4:    #include <conio.h>
5:    main ()
6:      {
7:          char ch;
8:          char pass[10];
```

```
9:        do
10:         {
11:           textcolor(WHITE);
12:           textbackground(BLUE);
13:           cprintf ("\r\n Enter Password:");
14:           textbackground(WHITE);
15:           cscanf ("%s",pass);
16:         } while ((strcmp(pass,"azab")) !=0);
17:      }
```

شكل ١٤-٣ برنامج كلمة سر لا تظهر على الشاشة

وعن هذا البرنامج نوضح مايلى :

يقوم البرنامج فى السطر رقم ١١و١٢ بتحديد لون الكتابة أبيض ولون الخلفية أزرق

- والسطر رقم ١٣ يطبع العبارة Enter Password بهذه الالوان.
- السطر رقم ١٤ يجعل لون الخلفية أبيض وبالتالى يتم الكتابة باللون الابيض على خلفية بيضاء فلا تظهر كلمة السر وهذا هو المطلوب.
- ثم يعاود البرنامج تغير الألوان وهكذا حتى يدخل المستخدم كلمة السر الصحيحة.

# تذكر

| عندما تريد | إتبع الآتى |
|---|---|
| عمل دوارة تعتمد على عداد معلوم بدايته ونهايته | إستخدم الدوارة for |
| عمل دوارة تعتمد على شرط غير معلوم | إستخدم الدوارة while |
| تنفيذ الدوارة أولا ثم الشرط | إستخدم الدوارة do....while |
| تنفيذ أكثر من عبارة داخل البلوك | ضع القوس } فى بداية البلوك و القوس { فى نهاية الدوارة |



# الفصل الرابع

# التفريع Branshing

*البرامج التي شرحناها حتى الآن نفذت بمفهوم تسلسلي (sequetnial) بمعنى أن البرنامج ينفذ أول تعليمة ثم التي تليها إلى أن ينتهي البرنامج , وفي هذا الفصل سوف نتعرف على جمل الشرط التالية :*

- *جملة الشرط* **if**
- *جمل* **if** *الشرطية المتداخلة*
- *جملة الشرط* **if ......else**
- *جملة الشرط* **if ....... else if**
- *التركيب* **switch,case,break,default & continue.**
- *المؤثر الشرطي ؟*
- *جملة التفريع غير المشروط* **goto**

التفريع يعنى تغير مسار البرنامج. والتفريع إما يكون مشروط كجملة if أو غير مشروط كجملة goto

## التفريع المشروط

### *جملة الشرط if*

تستخدم كلمة if لتنفيذ جملة أو أكثر حسب شرط معين

وأبسط صورة لجملة if هى :

```
if (condition)
   statement;
```

ومعناها اذا تحقق الشرط (Condition) نفذ الجملة التالية أما اذا لم يتحقق الشرط فلا تنفذ هذه الجملة وانتقل إلى التى تليها.

*أمثلة*

يقوم البرنامج الموجود بالشكل ١-٤ باستقبال حرف ثم يختبر هذا الحرف باستعمال جملة if وعلى أساسها يطبع البرنامج النتيجة

```
0:  /*Program Name : cs4_1.c  */
1:  #include <stdio.h>
2:  #include <conio.h>
3:  main ()
4:     {
5:         char ch;
6:         clrscr();
7:         printf("\n type letter ....:");
8:         ch=getche();
9:         if (ch=='y')
```

```
10:            printf ("you typed letter y ");
11:    }
```

شكل ١-٤ استعمال الجملة if

وفى هذا البرنامج فى السطر رقم ٨ تقوم الداله ( ) getch باستقبال حرف وتخزينه فى المتغير ch وفى السطر رقم ٩ ، ١٠ يختبر هل هذا الحرف هو الحرف y أم لا ، فاذا كان الحرف المدخل هو y يطبع البرنامج الرسالة you typed letter y ثم تنتهى والا ينتهى البرنامج بدون طباعة الرسالة

معنى هذا أن جملة if جملـه اعتراضيـه إذا تحقق شرطها يتم تنفيذ التعليمات المعتمدة على الشرط ثم باقى تعليمات البرنامج ،أمااذا لم يتحقق الشرط فلا تنفذ هذه التعليمات وينتقل التنفيذ إلى باقى جمل البرنامج.



يفرق مترجم لغة C بين الحروف الكبيرة والحروف الصغيرة ولحل هذة المشكلة هناك طريقتين

١ . استخدام المؤثر المنطقى OR والذى يرمز له بالعلامة || كما يلى :

```
if(ch=='y'||ch=='Y')
```

٢ . استخدام دالة التحويل إلى الحروف الكبيرة ()toupper مثل :

```
ch=toupper(ch)
```

أما البرنامج الموجود بالشكل رقم ٢-٤ فيسمح للمستخدم بكتابة مجموعة حروف وفى النهاية يطبع البرنامج عدد هذه الحروف وعدد الكلمات الموجودة فى هذه الحروف. وذلك بإستعمال جملة الشرط if.

```
0:  /*Program Name : cs4_2.c   */
1:  #include <stdio.h>
2:  #include <conio.h>
3:  main ()
4:    {
5:        int charcnt=0;
```

```
6:      int wordcnt=0;
7:      printf ("\n type charcters: ");
8:      while (  (ch=getche()) !='\r')
9:        {
10:          charcnt ++;
11:          if (ch==' ')
12:             wordcnt ++;
13:        }
14:      printf ("\n charcter count is %d",charcnt);
15:      printf ("\n word count is %d",wordcnt+1);
16:   }
```

شكل ٢-٤ برنامج عد الحروف والكلمات

وعن هذا البرنامج نوضح مايلي:

- فى السطر رقم ٧ يطبع البرنامج الرسالة type charcters
- يشتمل السطر رقم ٨ على الدوارة while وهى تختبر الحرف الذى استقبلته الدالة getche() وطالما أن هذا الحرف ليس مفتاح Enter يستمر البرنامج فى استقبال حرف جديد.
- فى السطر رقم ١٠ تزداد قيمة المتغير charcnt بمقدار واحد كلما استقبل حرف جديد وبالتالى يعبر المتغير charcnt عن عدد الحروف.
- السطر رقم ١١ يختبر ما إذا كان الحرف المدخل عبارة عن مسافة خالية أم لا فاذا كان الحرف المدخل مسافة خالية فهذا معناه أن المستخدم ضغط على مفتاح المسافات أى كتب كلمة جديدة وبالتالى يزداد عداد الكلمات بمقدار واحد ويظل هكذا حتى اذا ضغط المستخدم على مفتاح Enter ينهى البرنامج سطر ١٤ ، ١٥ يطبع عدد الحروف وعدد الكلمات التى أدخلها المستخدم و عند تنفيذ البرنامج ستحصل على النتيجة التالية :

```
type charcters : Allah the god of all world
charcter count is 27
word count is 6
```

اذا كان هناك أكثر من جملة نريد تنفيذها مع if لابد من فتح قوس } قبل مجموعة الجمل والقوس { في أخر الجمل كما يلى

```
if (condition)
   {
     statement1;
     statement2;
   }
```

## جمل if الشرطية المتداخلة

يمكن أن تتداخل جمل if فتأخذ الشكل التالى :

```
if (condition)
   if (condition)
     if (condition)
```

وهذا معناه اذا تحقق الشرط الاول انظر الى الشرط الثانى .. وهكذا

*مثال*

البرنامج الموجود بالشكل ٣-٤ تطبيق على تداخل جمل if

```
0:  /* Program Name : cs4_3.c   */
1:  #include <stdio.h>
2:  #include <conio.h>
3:  main ()
4:    {
5:      printf("type letters ....:");
6:      if (getch()=='n')
7:          if (getch()=='o')
8:              printf ("\n you typed no");
9:    }
```

شكل ٣-٤ جمل if المتداخلة

وفى هذا البرنامج تستقبل الدالـة getch() حـرف وتقارنـه مـع الحـرف n فـاذا كانت المقارنة صحيحة تنفذ الجملة التالية التى هى شرط آخر فـاذا تحقـق هـذا الشـرط (وهو أن الحرف الثانى o) تطبع الرسالة you typed no



مـاذا يحـدث لـو كـان الحـرف الأول أى حـرف غـير الحـرف n نفـذ البرنـامج وأدخل حرفاً آخر. ولاحظ النتيجة وحاول تفسيرها.

## الجملة الشرطية if ...... else

تستخدم لتنفيذ أحد إختيارين وتأخذ الصورة التالية :

```
if (condition)
   {
     statement1
   }
  else
   {
     statement2
   }
```

ومعناهـا اذا كـان الشـرط (Condition) صحيـح نفـذ الجملـة الاولـى (Statement1) والا نفذ الجملة الثانية statement2

وهذا يعنى أن تركيب if .. else يستخدم لتحديد اختيار واحد من اختيارين ولايمكن تنفيذ الاختيارين معا كما يحدث مع جملة if وحدها .

و البرنامج الموجود بالشكل ٤-٤ يستعمل التركيب if..else

```
0:  /*Program Name : cs4_4.c  */
1:  #include <stdio.h>
2:  #include <conio.h>
3:  main ()
4:    {
5:       char ch;
6:       printf("\n\n type character :");
```

```
7:      if ((ch=getche()) =='y')
8:          printf ("\n you typed letter y");
9:        else
10:         printf ("\n you typed another letter");
11:    }
```

شكل ٤-٤ استخدام if .. else

وعن هذا البرنامج نوضح ما يلى :

فى السطر رقم ٧ الدالة ()getche تستقبل حرف وتخزنه فى المتغير ch وتقارن جملة if اذا كان هذا الحرف هو حرف y أم لا. فإذا كان الحرف يساوى حرفy فهذا معناه أن الشرط تحقق وبالتالى تظهر الرسالة you typed letter y.

- وسطر ٩ else معناه وإلا
- وسطر ١٠ يطبع العبارة you typed another letter

والبرنامج الموجود بالشكل رقم ٥-٤ يقوم باستعمال التركيب if ..else مع دوارتين for ليعطى النتيجة الموجودة بالشكل رقم ٦-٤

```
0:  /* Program Name : cs4_5.c */
1:  #include <stdio.h>
2:  #include <conio.h>
3:  main()
4:   {
5:     int r,c;
6:     clrscr();
7:     for(r=0;r<20;r++)
8:       {
9:         printf("\n");
10:        for(c=0;c<40;c++)
11:          {
```

```
12:            if(r==c||c==40-r)
13:                printf("*");
14:            else
15:                printf(".");
16:        }    /* secound for */
17:    }        /* first for */
18: }           /* End of main() */
```

شكل ٥-٤استعمال التركيب if...else مع دوارتين

وعن هذا البرنامج نوضح ما يلى :

يستخدم هذا البرنامج فكرة استخدام الدوارات المتداخلة(Nested Loops) وهى عبارة عن دوارتى for لطباعة شبكة من النقط حيث تبدأ فى السطر رقم ٧ الدوارة forيتنفيذ ما بداخلها ٢٠ مرة ودوارة for أخرى فى السطر رقم ١٠ تقوم يتنفيذ ما بداخلها ٤٠ مرة وهو طباعة النقطة . وبالتالىنتيجة سطر رقم ٧ وسطر رقم ١٠ هو رسم شبكة من النقط. عبارة عن ٤٠ نقطة فى ٢٠ سطر

فى السطر رقم ١٢ اذا تحقق هذا الشرط يضع بدل النقطة الحرف * لتحصل على شكل ٦-٤

```
.*.....................................*
..*...................................*.
...*.................................*..
....*...............................*...
.....*.............................*....
......*...........................*.....
.......*.........................*......
........*.......................*.......
.........*.....................*........
..........*...................*.........
...........*.................*..........
............*...............*...........
.............*.............*............
..............*...........*.............
```

```
................*.........*................
.................*.......*.................
..................*.....*..................
...................*...*...................
....................*.*....................
```

شكل ٦-٤ نتيجة تنفيذ برنامج cs4-5.c

## الجملة الشرطية if ....... else if .

لتنفيذ خيار من مجموعة خيارات كمقارنة رقمين مثلا حيث يكون الرقم الأول أكبرمن أو يساوى أوأقل من الرقم الثانى. يوجد طريقتين ، الطريقة الأولى إستخدام ثلاث جمل if وفى كل جملة نضع أحد الشروط الثلاثة كالآتى :

```
i=5;
if (i<5)
   printf("i less than 5");
   if (i=5)
      printf("i equal to 5");
      if (i>5)
         printf("i greater than 5");
```

ويعيب تلك الطريقة أن البرنامج سيقوم بإختبار شروط if الثلاثة حتى وإن كان الشرط قد تحقق فى جملة if الثانية فهو لابد أن يختبر جملة if الثالثة لأن كل جملة من جمل if مستقلة بنفسها ويجرى تنفيذها على حدة مما يستهلك وقتا فى إختبار جمل شرطية لاداعى لإختبارها حيث نفذت إحداها بالفعل.

والطريقة الثانية تستخدم لتلافى ذلك العيب وفيها نستخدم الجملة الشرطية if...else if. وصيغتها كالآتى :

```
if (condition)
```

```
        statement1;
      else if (condition )
          statement2;
        else if (condition)
            statement3;
```

فى هذه الصيغه لا يتم إختبار جملة if الثانيه إلا إذا كانت جملة if الأولى غير صحيحه.

وفى هذه الحالة تصبح صيغة المثال السابق كالآتى :

```
i=5;
if (i<5)
     printf("i less than 5");
   else if (I=5)
        printf("i equal to 5");
      else if (i>5)
             printf("i greater than 5");
```

وهذا معناه تحديد اختيار من عدة اختيارات

والبرنامج الموجود فى الشكل ٧-٤ يستعمل التركيب if else.. if فى عمل مثال لالة حاسبة بسيطة حيث يقوم البرنامج بطباعة رسالة يطلب فيها رقمين وبينهما علامة حسابية وعند ادخال الرقمين والعلامة الحسابية يطبع البرنامج النتيجة ثم يسأل هل تريد الاستمرار ويستمر هكذا طالما تجيب بالحرف y نفذ البرنامج ثم تابع معنا الشرح

```
0:  /* Program Name : cs4_7.c  */
1:  #include <stdio.h>
2:  #include <conio.h>
3:  main ()
4:    {
5:       float num1,num2;
6:       char op,ch;
7:        do
```

```
8:          {
9:            printf ("\n Type num1,op,num2:");
10:           scanf ("%f %c %f",&num1,&op,&num2);
11:          if (op=='+')
12:                printf ("sum=%f",num1+num2);
13:            else if (op=='-')
14:                printf ("sub=%f",num1-num2);
15:            else if (op=='/')
16:                printf ("div=%f",num1/num2);
17:          printf ("\n again(y/n)");
18:        }while ((ch=getch() )=='y');
19:   }
```

شكل ٧-٤ برنامج يستخدم التركيب if...else if لعمل آلة حاسبة بسيطة

وعن هذا البرنامج لوضح ما يلى :

- السطر رقم ٩ يطبع العبارة :Type num1,op,num2 باستخدام الدالة printf().
- في السطر رقم ١٠ تستقبل الدالة scanf() ثلاث متغيرات هم رقم ، ومؤثر (علامة حسابية) ، ورقم.
- فى السطر رقم ١١ جملة if تقول اذا كان المؤثر (العلامة الحسابية) المدخل هى علامة الجمع + نفذ السطر رقم ١٢ وهو جمع الرقمين وطباعة النتيجة. وهكذا باقى السطور حتى السطر رقم ١٦وهى الحالات الأربعة للمؤثر.
- فى السطر رقم ١٨ الدالة getch() تستقبل حرف وتختبر جملة while هذا الحرف فإذا كان حرف y يعود التنفيذ لبداية الدوارة من أول السطر رقم ٩ وإلا انتهى تنفيذ البرنامج. وبالتالى يماثل هذا البرنامج آلة حاسبة بسيطة لعمليات جمع وطرح وقسمة وضرب ويتم الخروج من البرنامج بادخال أى حرف غير الحرف y

وعند تنفيذ البرنامج نحصل على النتيجة التالية :

```
Type num1,op,num2:45+66
sum=111.000000
again(y/n)
Type num1,op,num2:88+22
sum=110.000000
again(y/n)
Type num1,op,num2:95-8
sub=87.000000
again(y/n)n
```

## التفريع switch ........case

لاحظت فى المثال السابق أن إستخدام جملة if فى حالة تعدد الإختيارات لأكثر من إختيارين يمثل عبئا على المبرمج فى تتبع خطوات البرنامج ويسبب بطئا نسبيا فى تنفيذ البرنامج لذا إستخدمنا الجملة الشرطية if...else if ويمكن استعمال التفريغ switch ... case كبديل لجملة If ... else if وهى طريقة أسهل كما سنرى وتستخدم بالصيغة التالية

```
ch=getch();
switch (ch)
 {
 case '1':
    statement1;
    statement2;
    ....
    break;
 case '2':
    statement1;
    statement2;
```

```
        ....
        break;
    default:
        statement1;
        statement2;
         ...
      }
```

فى هـذا التركيب تختبر جملـة switch المتغير OP ثـم تفرض لـه مجموعـة حالات هذه الحالات تحدد بكلمة case ، ففى الحالة الاولى (: "case "a) إذا كـانت قيمة المتغير OP هى a يتم تنفيذ الجمل التالية والموجودة فى هـذه الحالـة ثـم الخـروج من تركيب .. switch بإستخدام كلمة break ، بالمثل الحالة الثانية وهكذا. أمـا إذا لـم تحقق حالة من الحالات يتجه التنفيذ إلى كلمـة default وهـى بمعنـى إذا لـم يتحقـق أى حالة من الحالات السابقة نفذ مايلى:

وننصح باستعمال هذا الاسلوب فى حالة اختيار حالة من مجموعة حالات .

والبرنـامج الموجـود بالشـكل ٨-٤ تعديـل لبرنـامج الالـة الحاسـبة السـابق والموجـود بالشكل ٧-٤ ولكن باستخدام التركيب switch ... case

```
0:/* Program Name : cs4_8.c */
1:   #include <stdio.h>
2:   #include <conio.h>
3:   main ()
4:     {
5:        float num1,num2;
6:        char ch,op;
7:        do
8:          {
9:           clrscr();
10:          printf ("\n Type num1 op num2:");
11:          scanf ("%f %c %f",&num1,&op,&num2);
```

```
12:      switch (op)
13:        {
14:          case '+':
15:            printf ("sum=%f",num1+num2);
16:            break;
17:          case '-':
18:            printf ("sub=%f",num1-num2);
19:            break;
20:          case '*':
21:            printf ("mul=%f",num1*num2);
22:            break;
23:          case '/':
24:            printf ("div=%f",num1/num2);
25:            break;
26:          default:
27:            printf ("\n unknowen operator..");
28:        }
29:      printf ("\n again (y/n):");
30:    }while ((ch=getch()) =='y');
31:  }
```

شكل ٨-٤ استخدام التركيب switch...case

وعن هذا البرنامج نوضح مايلي :

- في السطر رقم ١١ الدالة ()scanf تستقبل ثلاث متغيرات رقمين ، ومؤثر (علامة حسابية) ، و في السطر ١٢ الجملة Switch تختبر المتغير opوفي السطر رقم ١٤ كلمة case تحدد حالة ما اذا كان المؤثر (العلامة الحسابية) هو علامة الجمع + فاذا كان المؤثر هو علامة الجمع + ينفذ السطر رقم ١٥ وهو جمع الرقمين وطباعة النتيجة ثم الخروج من الاختيارات عن طريق break وهكذا السطور حتى السطر رقم ٢٨

- في السطر رقم ٣٠ دالة getch() تستقبل حرف وتختبره جملة while اذا كان y يعود التنفيذ مرة أخرى من أول السطر رقم ٩ والا انتهى تنفيذ البرنامج

ومن هذا البرنامج نجد أن التركيب case ... switch أكثر وضوحا من التركيب if..else if.

وعند تنفيذ البرنامج نحصل على النتيجة التالية

```
Type num1 op num2:55+55
sum=110
again (y/n):y
Type num1 op num2:100-50
sub=50
again (y/n):n
```

*مثال*

من التطبيقات المشهورة لاستخدام التفريع switch().....case هو استخدامه في قوائم الاختيارات (menu) كما في الشكل التالي :

```
MAIN MENU
        1-INTRODUCTION TO C-LANG.
        2-STRUCTUTE OF C-PROG.
        3-LOOPS
        4-DECISIONS.
        ENTER SELECTION:
```

ويحتوى شكل ٩-٤ على البرنامج المطلوب لإظهار هذه القائمة والتعامل معها

```
0:  /* Program Name : cs4_9.c  */
1:  #include <stdio.h>
1:  #include <conio.h>
2:  main()
3:      {
```

```
4:      int  se;
6:      do {
7:        clrscr();
8:        printf("\n      MAIN MENU");
9:        printf("\n         1-INTRODUCTION TO C-LANG.");
10:       printf("\n         2-STRUCTUTE OF C-PROG.");
11:       printf("\n         3-LOOPS");
12:       printf("\n         4-DECISIONS.");
13:       printf("\n        ENTER SELECTION:");
14:       scanf("%d",&se);
15        switch(se)
16:           {
17:             case 1:
18:                 printf("\n 1-INTRODUCTION TO C-LANG.");
19:                 break;
20:             case 2:
21:                 printf("\n  2-STRUCTUTE OF C-PROG.");
22:                 break;
23:             case 3:
24:                 printf("\n  3-LOOPS");
25:                 break;
26:            case 4:
24:                 printf("\n  4-DECISIONS");
25:                 break;
26:           default:
27:                 printf("\n\n unknowen selection");
28:                 break;
29:           }
30:        printf("\n\n again(y/n)==>");
31:     } while(getch()=='y');
32:  }
```

شكل ٩-٤ برنامج يستخدم التركيب switch...case

وعن هذا البرنامج نوضح مايلى:

- السطور من ٨ الى ١٣ تطبع القائمة الرئيسية
- السطر رقم ١٤ يستقبل رقم صحيح ويخزنه فى المتغير se
- فى السطر رقم ١٥ يختبر المتغير se ويحدد قيمته
- السطر ١٧ يختبر قيمة المتغير se فإذا كانت الرقم ١
- السطر ١٨ يطبع الرسالة التالية 1-INTRODUCTION TO C-LANG
- وهكذا حتى السطر رقم ٢٩ وهو آخر سطر فى تركيب Switclh ... Case
- السطر ٣٠ يطبع الرسالة <==again(y/n) والسطر ٣١ يستقبل حرف ويختبره فإذا كان حرف y تعيد الدوارة while تنفيذ البرنامج مرة أخرى وهكذا

## المؤثر الشرطى ? (Conditional operator)

هذا المؤثر يقوم مقام جملة if

ويأخذ الشكل الاتى :

```
var=(condition) ? num1:num2;
```

ومعناه

```
if condition is true  then var = num1
if condition is false then var = num2
```

أى اذا كان الشرط صحيح فإن var=num1 واذا كان الشرط غير صحيح فإن var=num2

وهو بديل لتركيب if الذى يأخذ الشكل الآتى :

```
if (condition)
        var=num1;
 else
        var=num2;
```

## التفريع غير المشروط

التفريغ غير المشروط معناه الانتقال إلى مكان محدد داخل البرنامج بدون شرط وتقوم جملة goto بهذا الغرض وتأخذ الشكل العام التالى :

Goto lb :

حيث lb متغير من نوع char يشير الى المكان المطلوب الانتقال اليه ولاننصح بإستخدام هذه الجمله لأنها تستخدم بكثرة مع اللغات الغير تركيبية مثل لغة البيسك أما في حالة اللغة التركبية (لغة C) فيفضل ستخدام الدوال لتغير مسار تنفيذ البرنامج.

*مثال :*

البرنامج الموجود بالشكل ١٠-٤ يستخدم جملة goto للتفريع غير المشروط داخل البرنامج وفيه نلاحظ أن البرنامج سيطبع كلمة Allah باستمرار لان التفريع غير مشروط أى لا يوجد شرط لانهاء تكرار التنفيذ فاذا رغبت فى انهاء البرنامج فيجب الضغط على Ctrl+C

```
0:  /*Program Name : cs4_10.c  */
1:  #include <stdio.h>
2:  main ()
3:     {
4:       char  RE;
5:  RE:
6:       printf("\t allah");
7:       goto RE;
8:     }
```

شكل ١٠-٤التفريع الغير مشروط باستخدام أمر goto



# الفصل الخامس

# إنشاء الدوال والماكرو

# Functions & Macros

شرحنا فى الفصل دوال الادخال والاخراج وهى دوال موجودة داخل اللغة بامكانك انشاء دوال لتسهيل عملك وتبسيط برامجك لتقوم بوظائف محددة على غرار تلك الموجودة داخل اللغة.

فى هذا الفصل ستعرف الموضوعات التالية :

- ما المقصود بالدالة
- مثال لدالة بسيطة
- أنواع الدوال
- معاملات الدوال
- الماكروز **MACROS** والفرق بينها وبين الدوال
- كيفية ربط أكثر من ملف لانشاء ما يسمى بالمشروع **PROJECT**

## المقصود بالدالة

الدوال التي استخدمناها في الفصول السابقة مثل ()printf أو ()scanf دوال مبنية في لغة C وهي دوال عامة يستطيع أي مبرمج استخدمها. من مزايا لغة C المرونة في الاستخدام ولذلك يمكنك انشاء دوال مثل الدوال القياسية الموجودة في صلب اللغة لتؤدى وظائف مختلفة أو مشابهة والدالة عبارة عن برنامج صغير (أو مجموعة تعليمات تؤدى غرض معين) يخصص لهذا البرنامج إسم ويتم استدعائه داخل الدالة الرئيسية ( ) main بهذا الاسم.

و يحقق إستخدام الدوال مزايا عديدة منها :

١. عدم تكرار التعليمات داخل البرنامج حيث يتم إنشاء الدالة مرة واحده ثم يتم استدعائها اكثر من مره عند الحاجة اليها

٢. بإستخدام الدوال يصبح البرنامج اكثر وضوحا حيث يأخذ البرنامج الشكل التركيبي فيصبح بالشكل الاتي :

```
#include<filename.h>
 functions declerations;
 main ()
  {
     function1_calling();
     function2_calling();
     ........
     ........
  }

function1_defination()
  {

  }
```

```
function2_defination()
  {

  }
```

وبهذا يصبح البرنامج كما ترى اسهل للفهم حيث يتكون من الدالة الرئيسيه ومن داخلها يتم إستدعاء مجموعه من الدوال وبالتالى يكفى أن تفهم عمل كل داله لفهم البرنامج كله.

٣. يمكن للمبرمج المتمرس انشاء مكتبه دوال خاصه توفر عليه إعاده كتابة البرامج فى كل مرة يحتاج اليها

## مثال لدالة بسيطة

يوضح البرنامج الموجود بالشكل رقم ( ١-٥) كيف يتم الاعلان عن الدالة واستدعائها وكيف تظهر تعليماتها داخل البرنامج.

```
0:  /*CS5_1.C*/
1:  #include <stdio.h>
2:  #include <conio.h>
3:  void line2 (void)
4:  main ()
5:     {
6:         clrscr();
7:         line2();
8:         printf ("***** Allah the god of all world ******\n");
9:         line2();  } /* END OF main()  function*/
10:        void line2 (void)
11:            {
12:              int j;
13:              for (j=0;j<=40;j++)
14:                  printf ("*");
```

```
15:          printf ("\n");
16:        }
```

شكل ٩-٥ مثال لدالة بسيطة

فى البرنامج الموجود بالشكل رقم (١-٥) أنشأنا داله بالإسم line2() وقد ظهرت كلمه line2() فى ثلاث مواضع :

*الموضع الأول* يسمى بالإعلان عن الدالة function decleration يكون ذلك قبل الدالة الرئيسيه main() كما فى السطر رقم ٣ ونلاحظ الفاصلة المنقوطة فى نهاية هذا الجزء لأنه اعلان، ويكون بالصورة ;void line2() وكلمه void هى نوع الدالة التى سوف نتكلم عنها فيما بعد بالتفصيل

*الموضع الثانى* داخل الدالة الرئيسيه : ويظهر فى أى مكان داخل الدلة الرئيسية ويسمى function caling أى استدعاء الدالة ويكون بالشكل ;line2() كما فى السطر رقم ٧ وسطر رقم ٩ وفيه يتم كتابه إسم الدالة فقط بدون نوع. واذا كان لها معاملات نكتب المعاملات

*الموضع الثالث* يكتب بعد إنتهاء الدالة الرئيسيه main().وهذا الجزء يسمى تعريف الدالة Function defination وفيه يتم كتابه محتويات الدالة وتبدأ فى البرنامج الذى بين أيدينا من السطر رقم ١٠ باسم الدالة ثم بالقوس } وكأنها برنامج صغير وبعد هذا القوس نبدأ كتابة تعليمات الدالة والدالة هنا عبارة دوارة for تقوم مع الدالة printf() بطباعة العلامة * ٤٠ مرة وعند استدعاء هذه الدالة يتم تنفيذ هذه السطور أى طباعة العلامة * ٤٠ مرة.

ويتضح ذلك من نتيجة التنفيذ التالية :

```
****************************************
***** Allah the god of all world ******
****************************************
```

ماذا تتخيل أن يكون شكل البرنامج السابق بدون دوال

يكون البرنامج كما فى الشكل رقم (٢-٥)

```
#include <stdio.h>
#include <conio.h>
 main ()
  {
      clrscr();
      for (j=0;j<=4;j++)
         printf ("*");
     printf ("\n");
     printf ("***** Allah the god of all world ******");
        for (j=0;j<=4;j++)
         printf ("*");
     printf ("\n");

  }
```

شكل رقم ٢-٥البرنامج بدون الدالة line 2()

فى البرنامج الموجود فى الشكل رقم ٢-٥ يبدو الفرق بين حجم هذا البرنامج والبرنامج الموجود بالشكل رقم ١-٥ بسيطا وذلك لأن حجم الدالة صغير إلا أن الفرق بين حالة استخدام الدوال من عدمه يتضح أكثر عند استخدام الدوال الكبيرة. وهكذا نرى أن استخدام الدوال يسهل قراءة البرنامج وفهمه ومراجعته

# أنواع الدوال Functions Types

في شرحنا للبرنامج الموجود بالشكل رقم ١-٥ عرفت أن الاعلان عن الدالة يتم بالصورة التالية :

void line2();

وعرفنا أن كلمه void هي أحد أنواع الدوال وهناك أنواع اخرى من الدوال نوضحها فيما يلي :

- دوال تعيد قيمة صحيحه int function
- دوال تعيد قيمة حقيقية float function
- دوال تعيد عبارة حرفية string function
- دوال تعيد حرف واحد char function
- دوال لا تعيد أي قيمة void function
- دوال تعيد قيمة من نوع structure وتسمى struct function

ولكن ماذا يعني أن نوع الدالة int أو float أو ..... الخ

لتوضيح ذلك انظر البرنامج الموجود بالشكل رقم (٣-٥) والشرح الوارد بعده.

```
0:  /*CS5_3.C*/
1:  #include <stdio.h>
2:  int sum(int a,int b);
3:  main ()
4:     {
5:        int z,x=10,y=40;
6:        z=sum(x,y);
7:        printf("\n\n z=%d",z);
8:     }
9:  int sum(int a,int b)
10:    {
```

```
11:      int s;
12:      s=a+b;
13:      return s;
14:    }
```

شكل ٣-٥ إنشاء دالة داخل البرنامج

وعن البرنامج الموجود فى الشكل رقم (٣-٥) نوضح ما يلى:

- فى السطر رقم ٢ تم الاعلان عن دالة بالاسم sum() وسبقت بالكلمة int وهى نوع الدالة وتقابل كلمة void فى البرنامج السابق ونلاحظ فى الاعلان عن الدالة وجود متغيرين بين الاقواس وهما معاملات الدالة .
- فى السطر رقم ٦ يتم استدعاء الدالة وبين أقواسها المتغيران x,y ويستخدمان كمعاملات للدالة ولابد من كتابة معاملات الدالة وذلك لاننا أعلنا عنها بهذه الصورة
- تشتمل السطور من رقم ٩ الى ١٤ على جمل الدالة نفسها ونوضحها فيما يلى:
- فى السطر رقم ٩ نعوض عن المتغير a بالقيمة الموجودة فى المتغير x (وهى ١٠) وبالمثل نعوض عن المتغير b بالقيمة الموجودة فى المتغير y (وهى ٤٠) وفى السطر رقم ١٢ نجمع محتويات كلا من المتغير a والمتغير b ونضع النتيجة فى متغير جديد هو s.
- وفى السطر رقم ١٣ نطلب اعادة محتويات المتغير s الى مكان استدعاء الدالة بإستخدام كلمة return وهو السطر رقم ٦ .وبهذا تفهم أن جملة z=sum(x,y) الموجودة بالسطر رقم ٦ تعادل الجملة z=s.

ولكن حتى الان لم يتضح معنى أن الدالة من نوع int . معنى نوع الدالة يتضح من القاعدة التى تقول أن نوع الدالة يتوقف على القيمة المرتجعة من الدالة . فاذا كانت القيمة المرتجعة int (صحيحة) كان نوع الدالة int (صحيحا) واذا كانت القيمة المرتجعة float (حقيقية) كان نوع الدالة float أما الدالة التى لا تعيد قيمة وبعبارة

أخرى لاتشتمل على جملة return فتكون من نوع void ومن أمثلتها الدوال الموجودة فى لغة C فمثلا دالة clrscr() تستخدم لمسح الشاشة فقط وبالتالى لا تعيد قيمة. ولذلك فهى تنتمى الى النوع void.

**استدعاء الدالة**

يتم استدعاء الدوال اما بمعاملات أو بدون معاملات ، وتكون بدون معاملات كما فى الدالة line2() فى البرنامج الموجود بالشكل رقم ١-٥ وبدون معاملات معناه عدم كتابة قيم بين أقواس الدالة ، ويرجع ذلك الى كيفية الاعلان عن الدالة وكيفية تصميمها فالدالة line2() تم الاعلان عنها بأنها بدون معاملات ولذلك عند كتابة سطور الدالة لم تستقبل الدالة معاملات من الدالة الرئيسية وبتم إستعمال المعاملات كما فى الدالة sum() الموجودة فى البرنامج الموجود بالشكل رقم ٣-٥ فقد تم الاعلان عنها على أنها ستستقبل معاملات

ويوضح البرنامج الموجود بالشكل رقم (٤-٥) فكرة الاستدعاء بمعاملات و البرنامج عبارة عن تعديل الدالة line2() الموجودة فى البرنامج ١-٥ مع تغير اسمها الى line3() و اعطائها معامل من نوع int.

```
0:   /*CS5_4.C*/
1    #include <stdio.h>
2:   #include <conio.h>
3:   void line3 (int no);
4:   main ()
5:       {
8:               clrscr();
9:               line3(30);
10:              printf ("* Allah the god of all world *\n");
11:              line3(70);
12:      }  /* END OF main()  function*/
```

```
13:  void line3 (int no)
14:    {
15:         int j,no;
16:         for (j=0;j<=no;j++)
17:              printf ("*");
18:         printf ("\n");
19:    }
```

شكل ٤-٥ استدعاء الدالة بمعاملات

وفى هذا البرنامج تم الإعلان عن الدالة ()line3 فى السطر رقم ٣ ونجد بين القوسين ( int no) ومعناه أن للداله معامل واحد من نوع صحيح وهو no ويقوم كلا من السطر رقم ٩ والسطر رقم ١١ إستدعاء الدالة ;()line3 ولكن كل مره يتم إرسال قيمه مختلفه للمعامل ففى المره الأولى فى السطر ٩ تكون القيمه هى ٣٠ وفى المرة الثانية فى السطر رقم ١١ تكون القيمة هى ٧٠ وبمعنى أننا عوضنا عن المتغير no أول مرة بالقيمة ٣٠ وبالتالى تطبع الدالة العلامة * ٣٠ مرة وثانى مرة. بالقيمة ٧٠ وبالتالى تطبع الدالة العلامة * ٧٠ مرة وهذا ما تلاحظه من نتيجة التنفيذ التالية :

```
******************************
* Allah the god of all world *
**********************************************************************
```

## استدعاء الدالة بمتغيرات

فى البرنامج السابق تم استدعاء الدالة ()line3 بمعاملات من نوع قيم ثابتة موجودة بالبرنامج نفسه. بالاضافة الى ذلك يمكن أن تكون هذه المعاملات متغيرات تستقبل قيمها من المستخدم أو من داخل البرنامج ، وهذا مفيد عندما نريد استقبال قيمه هذه المتغيرات من المستخدم بدلا من تحديدها داخل البرنامج. لانها تتغير حسب الحالة

وبالتالى تعطى مرونة فى التعامل مع البرنامج ويتضح ذلك من البرنامج الموجـود بالشـكل ٥–٥

```
0:  /*CS5_5.C*/
1   #include <stdio.h>
2:  #include <conio.h>
3:  void line4 (int no);
4:   main ()
5:     {
6:         int k;
7:         clrscr();
8:         do {
9:              printf ("\******* Enter Length of line(-ve to exit) :");
10:             scanf("%d",&k);
11:             line4(k);
12:          } while(k>0);
13:      }

14:   void line4 (int no)
15:      {
16:          int j;
17:          for (j=1;j<=no;j++)
18:               printf ("*");
19:          printf ("\n");
20:      }
```

شكل ٥–٥ إستدعاء الدالة بمتغيرات

فى هذا البرنامج تم الاعلان عن دالة بالاسم line4()

فى السطر رقم ١٠ يتم استقبال قيمة عبارة عن رقم صحيح وتخزينه فى المتغير k وفى السطر رقم ١١ يتم استدعاء الدالة بالقيمة المخزنة فى المتغير k وهذه القيمة المخزنـة

فى المتغير k تتغير حسب البيانات التى يدخلها المستخدم ويتضح من ذلك نتيجة التنفيذ التالية :

```
******* Enter Length of line(-ve to exit) :30
******************************
******* Enter Length of line(-ve to exit) :60
************************************************************
******* Enter Length of line(-ve to exit) :-5
```

## أمثلة مختلفة على أنواع الدوال

فيما يلى سنوضح أمثلة على أنواع الدوال من خلال ٣ برامج

*مثال لدالة من نوع Void*

البرنامج الموجود بالشكل رقم ٦-٥ يقوم بانشاء دالة بالاسم twobeep() تقوم هذه الدالة بعزف نغمة بتردد متغير وهذه الدالة من نوع void ولاننا لانحتاج من هذه الدالة أن ترجع لنا أى قيمة فليس بها جملة return.

```
0:  /*CS5_6.C*/
0:  #include <dos.h>
1:  #include <stdio.h>
2:  #include <conio.h>
3:  void twobeep(void);
4:  main ()
5:     {
6:        twobeep();
7:        printf ("\n WELCOME WITH C-PROGRAMMING :");
8:        twobeep();
```

```
9:        }

10: void twobeep(void)
11:       {
12:         int j;
13:         for (j=0;j<10000;j++)
14:           {
15:             sound(j*10);
16:             delay(5);
17:             nosound();
18:           }
19:       }
```

شكل ٦–٥ برنامج لدالة من النوع void

تشتمل السطور من ١٠ الى ١٩ على الدالة twobeep() وتستخدم دالتين من دوال لغة C الاولى sound() وتستخدم لاخراج نغمة بتردد يتوقف على الرقم المكتوب بين القوسين. والثانية delay() لعمل فاصل زمني بين النغمة والتي تليها وتتوقف مدة هذا الفاصل الزمني على الرقم المكتوب بين القوسين.

أكتب البرنامج ونفذه وتابع النتيجة وهي سماع صوت من سماعة جهاز الكمبيوتر

*مثال لداله من نوع int*

البرنامج الموجود بالشكل رقم ٧–٥ مثال لبرنامج يستعمل دالة من نوع int وفي هذه الدالة يتم ارسال قيمة صحيحة لها وتقوم الدالة بحساب مربع القيمة واعادة حاصل الضرب الى الدالة الرئيسية عند الاستدعاء وبالمثل دالة تحسب مكعب قيمة وتعيدها عند الاستدعاء وبالتالي تكون القيم المرتجعة في الدالتين من نوع صحيح وهو نوع الدالتين.

```
0:   /*CS5_7.C*/
1:   #include <stdio.h>
```

```
2:   int sqr(int a);
3:   int qup(int q);
4:   main ()
5:     {
6:        int  s,qu,no=10;
7:        s=sqr(no);
8:        qu=qup(no);
9:        printf("\n  squar of no=%d",s);
10:       printf("\n   qupic of no=%d",qu);
11:     }

12:  int sqr(int a)
13:     {
14:        int v1;
15:        v1=a*a;
16:        return  v1;
17:     }

18:  int qup(int q)
19:     {
20:        int v2;
21:       v2=q*q*q;
22:       return v2;
23:     }
```

شكل رقم ٧-٥ برنامج لدالة من النوع int

**نتيجة التنفيذ**

```
squar of no =100
qupic of no =1000
```

### مثال لدالة من نوع float

في الشكل رقم (٨-٥) مثال لبرنامج يحتوى على دالة من نوع float تقوم بجمع رقمين واعادة النتيجة الى الدالة الرئيسية عند الاستدعاء والقيمة المرتجعة قيمة حقيقية (float ) وبالتالي يكون نوع الدالة float

```
0:   /*CS5_8.C*\
1:   #include <stdio.h>
2:   #include <conio.h>
3:   float add(float x ,float  y);
4:   main ()
5:     {
6:        float  no1,no2;
7:        printf ("\n Enter no1,no2:");
8:        scanf  ("%f,%f",&no1,&no2);
9:        printf ("\n addation of Numbers is %f",add(no1, no2));
10:    }

12:  float  add (float  x,float  y)
13:    {
14:       float  yt;
15:       yt=x+y;
16:       return yt;
17:    }
```

شكل ٨-٥ برنامج لدالة من النوع float

وعند تنفيذ هذا البرنامج نحصل على النتيجة التالية

```
Ente no1,no2: 3.2,4.3
addation of squars is 6.5
```

## معاملات الدالة الرئيسية ()main

حتى الآن تلاحظ أننا نكتب الدالة الرئيسيه ()main بدون معاملات. وبالتالى نحصل على الصورة القابلة للتنفيذ من البرنامج وتنفذ بدون معاملات. بعبارة اخرى يكفى ان تكتب اسم البرنامج تحت محث نظام التشغيل DOS لاستدعائه للتنفيذ. ولتوضيح ذلك أكتب البرنامج الموجود بالشكل ٩-٥ وقم بتنفيذه ثم إخرج الى بيئة نظام التشغيل DOS ثم اكتب أمر dir نجد أن هذا الملف له ثلاث نسخ هى cs5-9.obj cs5-9.exe .cs5-9.c النسخة القابلة للتنفيذ هى cs5-9.exe ولذلك يكفى أن تكتب اسم البرنامج لتنفيذ البرنامج الا هناك حالات تتطلب ادخال معاملات لتوجه البرنامج لغرض معين.

فمثلا الأمر format يتطلب ذكر اسم المشغل المطلوب تشكيله بعد كتابة الامر وهنا اسم المشغل يعتبر معامل للبرنامج :

```
/*CS5_9.C*/
#include <stdio.h>
main ()
  {
   printf("\n Welcome Mr.");
  }
```

شكل ٩-٥ برنامج يستخدام دالة الطباعة

فكيف لنا أن ننفذ البرنامج السابق cs5_9 بنفس الاسلوب بحيث نرسل له معامل ويقوم البرنامج باستقبال هذا المعامل واستعماله ؟

نفرض أن المطلوب هو الحصول على النتيجة التالية :

**Welcome Mr : Mohamed**

فى هذه الحالة يجب أن نرسل للبرنامج معامل هو كلمة Mohamed بالاضافة الى اسم البرنامج ويشتمل البرنامج الموجود فى الشكل رقم (١٠-٥) على البرنامج اللازم لهذا الغرض

```
0:   /*CS5_10.C*/
1:   #include <stdio.h>
2:   main (int argc,char *argv[ ])
3:     {
4:        if(argc!=2)
5:          {
6:            printf("\n\n Error in Arguments");
7:            exit(1);
8:          }
9:        printf("\n Welcome Mr.%s",argv[1]);
10:    }
```

شكل ١٠-٥ برنامج لاستقبال معامل من خارجه عند التنفيذ

وعند النظر إلى هذا البرنامج لأول مرة تلاحظ اختلاف شكل الدالة () main عما تعودت عليه فى المرات السابقة ،حيث يظهر داخل أقواس الدالة الرئيسية معاملين : الاول هو int argc والثانى هو char *argv[ ] ، والاول هو متغير من نوع صحيح عادى ويمكن أن يأخذ أى اسم ولكننا اخترنا هذا الاسم لانه اختصار للكلمة argument count أى عدد المعاملات ولقد صممت الدالة الرئيسية main() بحيث تضع فى المعامل الأول عدد المتغيرات التى يكتبها المستخدم فمثلا : لوكتبت الصيغة

copy a:file1.ext b:

تجد أن هذا السطر به ثلاث متغيرات الأول اسم البرنامج copy ، والثانى هو المعامل الأول للبرنامج وهو إسم الملف a:file.ext، والثالث هو b: ، اذن عدد المتغيرات التى كتبها المستخدم ثلاثة ويخزن هذا الرقم فى المتغير argc وفائدته تحديد

عدد المتغيرات التى سيقبلها البرنامج ساعة التنفيذ وبهذه الطريقة نستطيع جعل البرنـامج يؤدى أكثر من وظيفة وكل وظيفة تتوقف على عدد المعاملات التى يكتبها المستخدم.

و بهذا الرقم تقوم الدالـة الرئيسـية بحجـز مصفوفـة مـن العبـارات الحرفيـة (أى مصفوفة عناصرها كلمات (Strings).

عدد هذه العناصر هو argc واسـم هـذه المصفوفـة هـو المعـامل الثـانى للدالـة main() وهو argv ويتم تخزين المتغيرات التى يدخلها المستخدم فى هذه المصفوفة.

ماذا لو أردت كتابة برنامج وأردت تحديد اسـم البرنـامج بحيـث لايسـمح لأحد بتنفيذ هذا البرنامج باسم أخر بالرغم من أى برنـامج تنفيـذى تسـتطيع تغير اسمه واستعماله كما تشاء ؟

ستجد الحل فى القرص المصاحب للكتاب تحت اسم solv1e.c.

والان نتابع شرح البرنامج

فى السطر رقم ٤ جملة if تقول اذا كان المتغير argc لا يساوى ٢ أى اذا كان عدد المعاملات التى أدخلها المستخدم لا يساوى ٢ أى اذا لم يكتب المستخدم الصورة المطلوبة لتنفيذ البرنامج وهى

**CS5_10 Mohamed**

تظهر العبارة التالية

**Error in Arguments**

وينتهى عمل البرنامج بالدالة exit(1)

وفى السطر رقم ٩ اذا أدخل المستخدم الصورة الصحيحة تظهر الرسالة التالية

**Welcome Mr. Mohamed**

والسؤال من أين أتى البرنامج بكلمة Mohamed ؟

الاجابة عندما يكتب المستخدم الصورة

**CS5_10 Mohamed**

يوضع اسم البرنامج CS5_10 فى العنصر الأول من المصفوفة أى Argr [0]
وكلمة Mohamed فى العنصر التالى من المصفوفة أى Argr [1]

## الماكرو (MACROS)

### ما المقصود بالماكرو ؟

هو مجموعة تعليمات تؤدى غرض معين ويشبه الى حد كبير الدالة ،ويتم انشائه مرة واحدة وبعد ذلك يمكنك استدعائه كلما احتجت اليه.

وقبل أن نسأل ما الفرق بينه وبين الدالة تعال بنا نرى أولا كيفية انشائه واستعماله ثم نناقش بعد ذلك الفرق ثم نوضح متى نستخدم الماكرو ومتى نستخدم الدالة.

### كيفية انشاء الماكرو

يتم ذلك باستعمال الكلمة #define وهذه الكلمة تسمى directive أو preprocessor ومعناها توجيه.

ولانشاء الماكرو تستخدم الصورة التالية :

**#define macro line**

وهى عبارة عن تعريف طرف بطرف مثل #define A 5 ومعناها عرف المتغير A بالقيمة 5

والبرنامج الموجود بالشكل رقم ( ١١-٥ ) يوضح لنا كيفية الاعلان عن الماكرو وكيفية إستعماله

```
0:  /*CS5_11.C*/
1:  #define sum(a,b) a+b
2:  #define mul(x,y) x*y
3:  #include <stdio.h>
4:  main ()
5:      {
6:        int v1=5,v2=10;
7:        printf("\n\n sum(v1,v2)=%d",sum(v1,v2));
8:        printf("\n\n mul(v1,v2)=%d",mul(v1,v2));
9:      }
```

الشكل١١-٥ الاعلان عن الماكرو واستعماله

وعند تنفيذ هذا البرنامج ستحصل على النتيجة التالية :

sum(v1,v2)=15
mul(v1,v2)=50

وعن هذا البرنامج نوضح ما يلى :

فى السطر رقم ١ استخدمنا كلمة define لتعريف ماكرو بالاسم sum ووظيفته استبدال المتغيرين a,b بالصورة a+b وبالمثل فى السطر رقم ٢ يستبدل المتغيرين x,y بنتيجة الضرب x*y ومعناها كلما قابل مترجم اللغة الطرف الأول للماكرو يستبدله بالطرف الثانى.

ونلاحظ فى هذا المثال أن المتغيرين a,b يمكن استبدالهما بأى متغيرين أو قيمتين داخل البرنامج واسم الماكرو هو الذى يحدد العملية التى يقوم بها الماكرو هل هى عملية جمع أم ضرب بناءً على المعادلة الموجودة فى الطرف الايمن من الماكرو

فى السطر رقم ٧ استخدمنا الماكرو sum لجمع متغيرين هما v1,v2 وبالمثل السطر رقم ٨.

***ما الفرق بين الماكرو وبين الدالة ومتى نستخدم الماكرو ومتى نستخدم الدالة؟***

من العروف أن التعامل مع البرنامج يمر بثلاث مراحل :

١. كتابة البرنامج وهذا ما نسميه source code ويخصص لملف المصدر الامتداد C.
٢. ترجمة البرنامج للغة يفهمها الحاسب ويسمى compilation ويخصص للملف الامتداد obj.
٣. ربط ملف object بمكتبات اللغة ليصبح قابل للتنفيذ وتسمى هذه العملية linking ويخصص للملف امتداد exe.

ويظهر الفرق بين الماكرو وبين الدالة فى هذه المراحل كما يلى

فى مرحلة الكتابة ليس هناك فرق بين الماكرو والدالة .فى مرحلة الترجمة (compilation). يتم تحويل تعليمات الدالة الى لغة الالة (object) وتنتظر مرحلة الربط ولاتنفذ الدالة الا فى مرحلة الربط. أما فى حالة الماكرو يتم استبدال الماكرو بنتيجة تنفيذ الماكرو فبالرجوع الى البرنامج الموجود بالشكل رقم ١١-٥ يتم استبدال الماكرو الموجود بالسطر رقم ٧ بنتيجة التنفيذ مباشرة أى يتم وضع القيمة ١٥ التى هى نتيجة تنفيذ الماكرو مكان sum(v1,v2) وبالتالى عندما تأتى مرحلة التنفيذ يجد البرنامج نتيجة تنفيذ الماكرو وهى ١٥ وبالمثل يمكن أن تفهم الماكرو الموجود فى السطر رقم ٨

وهكذا نرى أن للماكرو مزايا منها :

أنه بسيط في الانشاء بسيط في الاستعمال ويعطى في النهاية ملف تنفيذى أصغر من المقابل له باستعمال الدوال فإذا كانت العملية المطلوبة عمل داله لها بسيطة ويمكن كتابتها في سطر واحد نستعمل الماكرو أما إذا كانت تحتاج أكثر من سطر نستخدم الدالة.

## المشروع Project

إذا كانت لك خبرة بقاعد البيانات DBASE أو CLIPPER ، فأنت تعلم أن انشاء تطبيق متكامل يتم عن طريق عمل ملف رئيسى (برنامج رئيسى) وبرامج فرعية.

ويتم استدعاء البرامج الفرعية من البرنامج الرئيسى وذلك من داخل البرنامج الرئيسى بالأمر do أى تقسيم البرنامج الى برنامج رئيسى وبرامج فرعية .

## والسؤال كيف يتم ذلك فى لغة C ؟

*يتم ذلك بأكثر من طريقة*

اذا كان البرنامج الكلى صغيرا ولا يتطلب أكثر من ملف يكفى استعمال الدوال كما مر بنا فى البرامج السابقة أى ننشئ الدالة الرئيسية ()main وبداخلها يتم استدعاء الدوال الاخرى.

اذا كان البرنامج كبير بحيث يتطلب أكثر من ملف فهناك طريقتين لتنظيم ذلك

*الطريقة الاولى* هى طريقة تقليدية اجتهادية لاتستعمل كثيرا وهى أن نكتب البرنامج الرئيسى وبه الدالة الرئيسية فى ملف منفرد ونكتب البرامج الفرعية فى صورة دوال ونضع هذه الدوال فى ملف منفصل. ويتم استدعاء هذه الدوال من داخل الدالة الرئيسية ولكى يتم ربط الملف الثانى الذى به الدوال بالملف الاول نكتب اسم الملف الثانى مع

كلمة #include بالصورة #include <file2.c> فى بداية الملف الاول ولتوضيح ذلك نسوق المثال التالى :

فى هذا المثال ستجد ملفين الاول اسمه calc.c وهو الموجود بالشكل رقم ١٢-٥ والملف الثانى اسمه tools.c وموجود بالشكل رقم ١٣-٥

```
0:  /*Program Name CALC.C*/
1:  #include "tools.c"
2:  #include <stdio.h>
3:  int sum(int a,int b);
4:  int mul(int x,int y);
5:  main ( )
6:      {
7:          int no1=5,no2=10;
8:          printf("\n\n no1+no2=%d",sum(no1,no2));
9:          printf("\n\n no1*no2=%d",mul(no1,no2));
10:     }
```

شكل رقم ١٢-٥ ملف البرنامج الرئيسى calc.c

وتلاحظ فى بداية البرنامج الأول الموجود فى الشكل رقم ١٢-٥ التوجيه #include "tools.c" ومعناه اربط الملف tools.c مع هذ الملف فى مرحلة الترجمة وبالتالى تصبح الدوال الموجودة بالملف tools.c كأنها بالملف calc.c

```
0:  /*Program Name TOOLS.C*/
1:  int sum  (int a,int b)
2:      {
3:          int yt;
4:          yt=a+b;
5:          return yt;
6:      }
```

```
7:  int mul (int x,int y)
8:      {
9:        int k;
10        k=x*y;
11        return k;
21      }
```

شكل رقم ١٣-٥ ملف الدوال tools.c

*الطريقة الثانية* وهى الطريقة المتبعة فى البرامج الكبيرة وهى انشاء مشـروع(project) هذا المشروع عبارة عن برنامج رئيسى يشتمل على دالة رئيسية ( ) mainوبرامج فرعيـة تشتمل فقط على دوال ولاتحتوى على دالة () main ويتـم ذلـك مـن خـلال بيئة كتابـة برامج لغة C ولتطبيق ذلك على الملفين السابقينCALC.C,TOOLS.C.

اتبع الخطوات الاتية

١. افتح ملف البرنامج calc.c (راجع شكل ١٢-٥)
٢. احذف السطر رقم ١ من البرنامج ثم احفظ الملف
٣. اضغط مفتاحى alt+p تظهر قائمة project
٤. من قائمة project اختر new تظهر نافذة صغيرة تكتب فيها اسم المشروع
٥. اكتب اسم لهذا المشروع وليكن main ثم اضغط مفتاح الادخال
٦. اضغط مفتاح insert تظهر قائمة بها اسماء الملفات الموجودة عندك
٧. اختر الملف tools.c ثم الملف calc.c
٨. اضغط مفتاح Esc بعد الانتهاء من اختيار ملف المشروع
٩. اضغط مفتاحى Ctrl+F9 لتنفيذ المشروع

وبهذ تحصل على ملف تنفيذى باسم المشروع وهو main

### مثال

نختم هذا الفصل ببرنامج متكامل يجمع معظم المفاهيم التى شرحناها حتى الآن (انظر شكل رقم ١٤-٥ ) وهذا البرنامج يقوم بطباعة قائمة اختيارات على الشاشة عبارة عن عمليات الجمع والطرح والقسمة والضرب ويطلب من المستخدم تحديد اختيار وعندما يقوم المستخدم بتحديد اختيار يقوم البرنامج بتنفيذ هذا الاختيار.

اكتب البرنامج أو فتحه من القرص المصاحب للكتاب ثم نفذه وشاهد نتيجة التنفيذ.

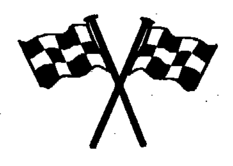
```
/*CS5_14.C-*/
#include <stdio.h>
#include <conio.h>
void add(void);
void sub(void);
void mul(void);
void div(void);
main ()
  {
    char ch='0';
    while (ch!='5')
     {
   /* draw menu */
   printf ("\n*******************************************************\n");
   printf (" **************** main menu *************************\n");
  printf ("\n (1)addation...");
  printf ("\n (2)sub..........");
  printf ("\n (3)mul..........");
  printf ("\n (4)Div..........");
  printf ("\n (5)exit.........");
  printf ("\n\n Enter selection :");
  ch=getch();
```

```
switch (ch)
   {
    case '1':
      add();
      break;
    case '2':
      sub();
      break;
    case '3':
      mul();
      break;
    case '4':
      div();
      break;
    case '5':
      ch='5';
      break;
    default:
      printf ("\n unknowen operator");
    }
   }
 }

void add();
   {
    float no1,no2;
    char op;
    clrscr();
    printf ("\n enter no1,op,no2");
    scanf ("%f%c%f",&no,&op,&no2);
    printf ("\n sum= %f",no1+no2);
   }
```

```
void sub()
    {
      float no1,no2;
      char op;
      clrscr();
      printf ("\n enter no1,op,no2");
      scanf ("%f%c%f",&no1,&op,&no2);
      printf ("\n sub= %f",no1-no2);
    }
void mul()
    {
      float no1,no2;
      char op;
      clrscr();
      printf ("\n enter no1,op,no2");
      scanf ("%f%c%f",&no1,&op,&no2);
      printf ("\n mul= %f",no1*no2);
    }
void div()
    {
      float no1,no2;
      char op;
      clrscr();
      printf ("\n enter no1,op,no2");
      scanf ("%f%c%f",&no1,&op,&no2);
      printf ("\n div= %f",no1/no2);
    }
```

شكل رقم ١٤-٥ برنامج قائمة اختيارات يستخدم الدوال



# الفصل السادس

# المصفوفات ARRAYS

- في هذا الفصل نتناول الموضوعات التاليه
- معنى المصفوفات وأنـواعها
- كيفيه التعامل مع المصفوفة ذات البعد الواحد
- المصفوفة ذات بعدين
- ارسال مصفوفة إلى داله كمعامل
- المصفوفات وسلسلة الحروف (الكلمـات) & Arrays String
- دوال العبارات الحرفية string functions
- أوامر المترجم preprocessors .

## معنى المصفوفات

تنقسم البيانات إلى بيانات حرفية (char) وبيانات رقمية (int) وبيانات حقيقيه (float) وتسمى هذه الأنواع (int,float,char) بالأنواع الرئيسيه للبيانات ، حيث لايمكن تجزئتها أقل من ذلك.

ولكن هناك أنواع أخرى من البيانات تسمى بالأنواع المشتقة (Drived data types) من هذه الأنواع المصفوفات Arrays. تعرف المصفوفة بأنها مجموعه من العناصر تنتمي إلى نوع واحد. ويخصص لها اسم واحد وتنقسم المصفوفات الى مصفوفات ذات بعد واحد ومصفوفات ذات بعدين

*والمصفوفة ذات البعد الواحد مثل :*

**A=[3 4 5 7 9]**

وتسمى مصفوفة ذات بعد واحد لأنها تتكون من صف واحد أو عمود واحد، وفيها حرف A هو اسم المصفوفة ،والارقام هي عناصر المصفوفة ويتم الاشارة الى كل عنصر برقم العنصر أى بترتيبه داخل المصفوفة على أن يبدأ العد بالرقم صفر كما يلى

العنصر A[0] يساوى ٣ و العنصر A[1] يساوى٤ والعنصر A[2] يساوى ٥ وهكذا ..

*والمصفوفة ذات البعدين تأخذ الشكل التالي :*

$$C = \begin{bmatrix} 5 & 4 & 2 \\ 7 & 1 & 7 \\ 2 & 9 & 5 \end{bmatrix}$$

وتسمى هذه المصفوفة ٣ × ٣ أى ٣ صفوف و ٣ أعمده ويتم الاشارة الى عناصر المصفوفة برقم الصف ورقم العمود الذى يقع عندهما العنصر كما يلى.

العنصر C [0] [0] يساوى ٥

العنصر C [0] [1] يساوى ٤

العنصر C [0] [2] يساوى ٢

العنصر C [2] [1] يساوى ٩

وهكذا ..

والخلاصة أن المصفوفة هى مجموعه من العناصر سواء ذات بعد واحد أو بعدين بشرط أن تكون جميع العناصر من نوع واحد وفيما يلى سنوضح كيفية الاعلان عن المصفوفة وكيفية التعامل مع عناصرها

## المصفوفة ذات البعد الواحد

البرنامج الموجود بالشكل رقم (١-٦) يوضح التعامل مع المصفوفة ذات البعد الواحد وفيه يتم الاعلان عن المصفوفة واستقبال عناصر المصفوفة من المستخدم واضافة قيمة صحيحة الى كل عنصر من عناصر المصفوفة ثم طباعة عناصر المصفوفة كما يتضح ذلك من نتيجة التنفيذ

```
0:  /*Program Name CS6_1.C*/
1:  #include<stdio.h>
2:  main ()
3:     {
4:      int A[10] ;
5:      int  i;
6:      for  (i=0;i<10;i++)
7:         {
8:            printf (" \n A[%d]=",i);
9:            scanf("%d",&A[i]);
```

```
10:              A[i]= A[i]+5;
11:           }
12:        for (i=0; i<10; i++)
13:           printf (" \n A[%d]=%d",i,A[i]);
14:    } /* main () function ----- */
```

شكل رقم (١-٦) التعامل مع مصفوفة ذات بعد واحد

وعند تنفيذ هذا البرنامج نحصل على النتيجة التالية :

```
A[0]=5
A[1]=7
A[2]=56
.......
......
A[0]=10
A[1]=12
A[2]=61
............
............
```

وعن هذا البرنامج نوضح مايلي :

- في السطر رقم ٤ اعلان عن مصفوفة عدد عناصرها ١٠ عناصر ونوع هذه العناصر int واسم هذه المصفوفة هو الحرف A وطريقة الاعلان عن المصفوفة بسيطة كما لو كنت تعلن عن متغير واحد ولكنك تضيف عدد عناصر المصفوفة إلى هذا المتغير .
- في السطر رقم ٦ إستخدمنا دوارة for لاستقبال عناصر المصفوفة ولاحظ استخدام دوارة for مع المصفوفة فاذا لم تستعمل دوارة for وكان عدد عناصر المصفوفة ١٠ عناصر فلابد من كتابة السطرين التاليين عشرة مرات لاستقبال عناصر المصفوفة

```
printf ("\n A[0] =" );
scanf("%d",& A[0] );
```

وهكذا - عشرة مرات - وهذا غير معقول لذلك نستخدم دوارة for كما يلى :

```
for (i=0;i<10;i++)
  {
    printf("\n A[%d]=",i)
    scanf("%d,& A[i] );
  }
```

لابد من استعمال الدوارة for مع المصفوفات

- فى السطر رقم ٨ دالة ( ) printf تطبع العبارة = A[0]
- فى السطر رقم ٩ دالة scanf() تستقبل الرقم الذى يدخله المستخدم وتخزنه فى عنصر المصفوفة A[i].

ونظرا لان المتغير i يبدأ بالقيمة صفر فان القيمة المدخلة تخزن فى العنصر الأول من المصفوفة والمشار اليه بالصورة A [0] ثم تزداد قيمة i وبالتالى تتوالى عناصر المصفوفة و تمتلىء بالترتيب



يشار لأول عنصر فى المصفوفة بالرقم صفر هكذا A[0].

فى السطر رقم ١٢ دوارة for أخرى لطباعة عناصر المصفوفة بعد اضافة الرقم ٥ الى كل عنصر

## اعطاء قيم ابتدائيه لعناصر المصفوفة

من الممكن الاعلان عن المتغير واعطائه قيمة ابتدائية بالشكل ا لاتى

```
int A=5;
```

وهذا اعلان عن متغير صحيح وفى نفس الوقت اعطاءه قيمة ابتدائية

وبنفس الاسلوب يمكن الاعلان عن المصفوفة واعطائها قيم ابتدائية كما يلى

```
int A[3] = {5,7,9} ;
char name [10] = { 'c', 'b' , 't' , 'r', -----} ;
```

وهذا معناه اعطاء قيم ابتدائية لعناصر المصفوفة وهو الافضل كلما استطعت ذلك حتى لا يقوم البرنامج بتخزين قيم عشوائية من الذاكرة فى عناصر المصفوفة وحتى لاتطبع قيم ليس لها معنى

## المصفوفة الغير محددة العدد

المقصود بها هو عدم تحديد عدد العناصر فى حالة الاعلان وتأخذ الصورة الاتية

```
int A[]={3,4,5} ;
```

أو

```
char name [ ]= " abdef ";
```

وتحديد عدد عناصر هذه المصفوفة فى هذه الحالة يتم من خلال المترجم عن طريق عد العناصر فى الطرف الايمن وحجز مصفوفة بهذا العدد

فمثلا ; int A[]={3,4,5} : معناه أن المصفوفة [ ] A عدد عناصرها ٣ عناصر وهكذا

وهذا لايصلح الا اذا كنت ستعطى عناصر المصفوفة قيم ابتدائية ولكن لا يصح أن تعلن عن مصفوفة غير محددة العدد ثم تستعملها فى استقبال قيم من المستخدم فمثلا لايصح أن تقول [ ]int a ثم تستقبل عناصر المصفوفة a من المستخدم.

*مثال*

يقوم البرنامج الموجود بالشكل (٢-٦) باستقبال الاسم ثم يطبعه بحيث تكون الحروف معكوسه كما يظهر ذلك من نتيجة التنفيذ.

```
0:   /*Program Name CS6_2.C*/
1:   #include <stdio.h>
2:   main ( )
```

```
3:    {
4:      int  i, c=0;
5:      char name [20] ;
6:      clrscr ( );
7:      printf (" \n Enter your name :")
8:      while (name [c]= getche ( ) ) !=' \r' )
9:          c++ ;
10:     printf("\n your name in reverse is :");
11:     for (i=c ;i>= 0; i--)
12:         printf ("%c", name [i] );
13  }
```

شكل رقم ٢-٦ استقبال اسم وعكس حروفه

وعن هذا البرنامج نوضح مايلى :

في السطر رقم ٥ اعلان عن مصفوفة حروف عدد عناصرها ٢٠ كحد أقصى ويشتمل السطر رقم ٨ على الدالة ()getche التى تستقبل حرف وتخزنه فى عنصر المصفوفة [name[c و c لها قيمة ابتدائية صفر اى تخزنه فى أول عنصر وتختبر جملة while هذا الحرف فاذا لم يكن هو مفتاح الادخال تستمر فى تنفيذ الجمل التالية لها

السطر رقم ٩ يزيد قيمة العداد c بمقدار واحد بالصورة ++c ويظل البرنامج يستقبل حرف ويزيد قيمة العداد c بمقدار واحد كلما كتب المستخدم حرفا وطالما لم يضغط المستخدم على مفتاح الادخال Enter

لاحظ أن الدالة ()getche لاتحتاج الى ضغط مفتاح Enter لذلك يستمر الاستقبال فى سطر واحد كعبارة حرفية.

- فى السطر رقم ١١ الدوارة for تبدأ من آخر عنصر تم ادخاله وهو آخر قيمة للمتغير c وتنتهى عند أول عنصر ورقمه صفر.

- فى السطر رقم ١٢ يتم طباعة عناصر المصفوفة بالعكس وذلك لاننا بدأنا من أخر عنصر ادخل حتى أول عنصر

وعند تنفيذ البرنامج تحصل على النتيجة التالية :

```
Enter your name:SAMY
your name in reverse is:YMAS
```

## المصفوفة ذات البعدين

هى المصفوفة التى ترتب عناصرها فى شكل صفوف واعمدة ويتم الاعلان عنها بالشكل التالى :

```
int A [5] [10] ;
```

وهذا معناه أن المصفوفة A. مصفوفة ذات بعدين ،٥ صفوف و ١٠ اعمدة ويتم الاشارة الى العنصر برقم الصف ورقم العمود

ويجب الانتباه الى أنه عندما تستخدم مصفوفة لابد من استعمال دوارة for ويتضح ذلك من المثال الذى ذكرناه فى المصفوفة ذات البعد الواحد وأما فى حالة المصفوفة ذات البعدين فلابد من استعمال ما يسمى بالدوارات المتداخله nested loops.

وهذا مانراه من خلال البرنامج الموجود بالشكل رقم (٣-٦) حيث يقوم باستقبال مجموعة قيم ويخزنها فى مصفوفة ذات بعدين ثم يقوم بطباعة هذه القيم فى شكل مصفوفة كما فى الشكل رقم (٤-٦)

```
0:  /*Program Name CS6_3.C*/
1:  #include <stdio.h>
2:  main ( )
3:     {
4:        int  r,c ;
5:        int A[3] [4] ;
```

```
6:      for (r=0;r<3; r++)
7:          {  printf ("\ n");
8:            for (c=0;c<4;c++)
9:              {
10:               printf ("\t A[%d] [%d]=" ,r,c);
11:               scanf ( "%d",& A[r] [c] ) ;
12:             }
13:         }
14:     for (r=0; r<3;r++ )
15:         {
16:           printf (" \n");
17:           for (c=0; c<4; c++ )
18:            printf (" \t %d ", A[r] [c] ) ;
19:         }
20:   }
```

شكل ٣-٦ التعامل مع مصفوفة ذات بعدين

وعن هذا البرنامج نوضح ما يلى :

- يبدأ البرنامج في السطر رقم ٥ بالاعلان عن مصفوفة ذات بعدين وذلك بالشكل int A [3] [4] وهى مصفوفة تتكون من ٣ صفوف و ٤ أعمدة من نوع صحيح.
- فى السطر رقم ٦ الدوارة for للتكرار ٣ مرات والسطر رقم ٨ للتكرار ٤ مرات وبالتالى كل قيمة للمتغير r تقابلها ٤ قيم لمتغيير C وهو ما تم دراسته فى الفصل الخاص بالدوارات.
- فى السطر رقم ١١ الدالة scanf() تستقبل العنصر رقم [r][c] وهو يتدرج من [0] [0] ، [1] [0] ، [2] [0] حتى يستقبل جميع عناصر المصفوفة.

ويشتمل كلا من السطر رقم ١٤ والسطر رقم ١٧ على دوارة for وهى تستخدم نفس الفكرة السابقة والسطر رقم ١٨ لطباعة عناصر المصفوفة.

وعند تنفيذ هذا البرنامج نحصل على النتيجة التالية :

```
A[0] [0]=55
A[0] [1]=66
A[0] [2]=77
A[0] [3]=88
A[1] [0]=99
A[1] [1]=44
A[1] [2]=33
A[1] [3]=22
..................
..................
    55 66 77 88
    99 44 33 22
    11 22 43 54
```

شكل (٤-٦) نتيجة استقبال وطباعة مصفوفة ذات البعدين

## إعطاء قيم ابتدائية للمصفوفة ذات بعدين :

كما يمكن اعطاء قيم ابتدائية للمصفوفة ذات البعد الواحد يمكن كذلك اعطاء قيم ابتدائية للمصفوفة ذات البعدين ويكون بالشكل الاتى :

```
int A[ 3] [4] = {
             { 4,5,7,8},
             {3,2,4,5},
             {7,9,8,1} } ;
```

وفى هذا الشكل يأخذ العنصر رقم 0،0 القيمة4، والعنصر رقم 0,1 القيمه 5 وهكذا

*مثال : (برنامج ورقة رسم بيانى)*

البرنامج الموجود بالشكل رقم (٥-٦) يطبع مصفوفة من النقط (٠) على الشاشه ثم يستقبل من المستخدم قيمة الصف والعمود ثم يوقع هذه النقطه على الشاشـه

فى شكل الحرف*. فإذا أدخل المستخدم قيمه سالبه ينتهى البرنامج وهذا البرنامج يمـاثل ورقة رسم بيانى كما يظهر ذلك فى نتيجة التنفيذ الموجودة بالشكل رقم (٦-٦).

```
0:  /*Program Name CS6_5.C*/
1:  #define rows 10
2:  #define cols 70
3:  main ()
4:   {
5:      int x,y;
6:      char matrix[rows][cols];
7:      for (y=0;y<rows;y++)
8:          for (x=0;x<cols;x++)
9:              matrix[y][x]='.';
10:
11:     do
12:       {
13:         clrscr();
14:         for (y=0;y<rows;y++)
15:            {
16:              printf ("\n");
17:              for (x=0;x<cols;x++)
18:                  printf("%c",matrix  [y][x] );
19:              printf ("\n");
20:            }
22:         printf ("\n Enter cordantes in form x,y -v to exit :");
23:         scanf ("%d,%d",&y,&x);
24:         matrix  [y][x] ='*';
25:      }while(y>0);
26:  }
```

شكل رقم (٥-٦) برنامج ورقة رسم بيانى

وعن هذا البرنامج نوضح ما يلى :

- يبدأ البرنامج فى السطر رقم ٦ بالاعلان عن مصفوفة ذات بعدين ولكن من نوع حرف char لاننا سوف نملأ عناصرها بحروف
- فى السطر رقم ٧و٨ دوراتين for لاننا نستعمل مصفوفة ذات بعدين
- السطر رقم ٩ يملأ عناصر المصفوفة بالنقطة .
- فى السطر رقم ١١ تبدأ دوارة do..while ثم مسح الشاشة بـ clrscr()
- السطور رقم ١٤و١٧و١٨ لطباعة المصبفوفة السابقة التى سبق ملء عناصرها بالنقطة. وبالتالى سيتم طباعة شبكة من النقط عبارة عن ١٠ صفوف و٧٠ عمود.
- فى السطر رقم ٢٣ يتم استقبال رقمين من المستخدم هما الاحداثيات المطلوب توقيعها. وفى السطر رقم ٢٤ يتم وضع العلامة * فى العنصر الذى يقابل الاحداثيات التى أدخلها المستخدم فمثلا اذا أدخل المستخدم الرقمين ٧،٥ فيوضع فى العنصر رقم [٧] [٥] العلامة * ثم تعيد جملة while التشغيل الى السطر رقم ١٣ حيث تمسح الشاشة وتعيد رسم الشبكة بالوضع الجديد وهو أن العنصر المقابل لاختيار المستخدم به العلامة * فيرسمها وبالتالى توقع على الرسم وهكذا حتى نحصل على نتيجة التنفيذ كما فى الشكل رقم ٦-٦

```
.......................................................................
..............................*........................................
....... ...........................*...................................
.........................*.............................................
.......*........................*......................................
..........*........................*...................................
............*..........................................................
.....................*.................................................
```

```
.......................................................................
Enter cordantes in form x,y -v to exit :
```

شكل رقم ٦-٦ورقة رسم بيانى

## مصفوفة العبارة الحرفيه Arroy of string

تستخدم كلمة سلسلة حروف مقابل لكلمة string وقد ذكرنا من قبل أن طريقة الاعلان عن سلسلة الحروف string هى ;char name[10] ومعناها مصفوقه من الحروف

ولكن ما معنى الاعلان التالى:

```
char name[5][10]
```

معناه مصفوفة حرفية عدد صفوفها ٥ وعدد اعمدتها ١٠أى ٥ صفوف كل صف عبارة عن كلمة لايتعدى عدد حروفها عن ١٠ حروف

وعلى ذالك يمكن أن تأخذ هذه المصفوفة قيمه بالشكل التالى

```
char name [5] [10]= {
                    {" string1"} ,
                    {" string2"} ,
                    {" string3"} ,
                    {" string4"} ,
                    {" string5"}
                              } ;
```

فى هذه المصفوفة يمكن التعامل مع حرف معين كما يمكن التعامل مع صف كامل على انه سلسلة حروف String ولذلك تسمى هذه المصفوفة array of stringأو مصفوفة حرفية.

والبرنامج الموجود فى الشكل رقم ٧-٦ يشتمل على مصفوفة عبارة عن مجموعة كلمات وفيه يطلب البرنامج من المستخدم ادخال كلمة سر فاذا كانت هذه

الكلمة أحدى الكلمات الموجودة في مصفوفة الكلمات سمح له البرنامج بتكملة العمل والا استمر البرنامج في دوارة do..while حتى يدخل المستخدم احدى الكلمات الموجودة بالمصفوفة

ويظهر ذلك في نتيجة التنفيذ الموجودة في الشكل رقم ٨-٦

```
0:  /*Program Name CS6_7.C*/
1:  #include <stdio.h>
2:  main ( )
3:    {
4:        int i; char name2 [10] ; int i=0 ;
5:        char name [5] [10] = {
6:                              {" ahmed   "} ,
7:                              {"mohamed"} ,
8:                              {"samy      "} ,
9:                              {" hamdy    "} ,
10:                             {"nabil      "} ,
11:                                        } ;
12:      do {
13:          clrscr ( );
14:          printf ("\n Enter your name :") ;
15:          scanf ( "%s", name2) ;
16:          for ( i=0; i<5 ; i++ )
17:            {
18:              if ( strcmp (name2, name [i] == 0 )
19:               t=1 ;
20:            }
21:      }while (t== 0);
22:  }
```

شكل رقم ٧-٦ مصفوفة العبارات الحرفيه

وعن هذا البرنامج نوضح ما يلي :

يبدأ هذا البرنامج فى السطر رقم ٥ بالاعلان عن مصفوفة حرفية وعطائها قيم ابتدائية عبارة عن مجموعة كلمات وذلك بالاسم name وتتسع المصفوفة لعدد ٥ اسماء كل اسم لايتعدى ١٠ حروف. وفى السطر رقم ١٢ يبدأ التكرار بـ do ثم مسح الشاشة ثم رسالة لاستقبال الاسم. فى السطر رقم ١٥ الدالة scanf() تستقبل سلسلة حروف وتخزنها فى المتغير name2 و يشتمل السطر رقم ١٦ على الدوارة for ووظيفتها تكرار المقارنة ٥ مرات ويستخدم السطر رقم ١٨ الدالة strcmp() للمقارنة بين الاسم الذى ادخله المستخدم والاسماء المخزنة فى المصفوفة فاذا كانا متساويين أى الفرق بينهما صفر اصبحت قيمة المتغير t تساوى القيمة ١ وبالتالى لن يتحقق شرط التكرار الموجود فى while أما اذا كانا غير متساويين فان while تعيد التنفيذ الى السطر رقم ١٣ وبالتالى يتم مسح الشاشة وقبول كلمة اخرى وهكذا حتى يدخل المستخدم كلمة من الكلمات الموجودة فى المصفوفة كما يظهر من تنفيذ البرنامج التالية :

```
Enter your name :aly
Enter your name :samah
Enter your name :hany
Enter your name :samy
```

شكل رقم ٨- ٦نتيجة التنفيذ

## ارسال مصفوفة للداله كمعامل

من الامور المهمه استعمال المصفوفة كمعامل من معاملات الداله ويتم توضيح ذلك فى البرنامج الموجود فى الشكل رقم ٩-٦.

وفى هذا البرنامج يتم الاعلان عن مصفوفة اسماء وكذلك الاعلان عن دالة أحد معاملاتها مصفوفة ويتم استدعاء الدالة مع ارسال مصفوفة حرفية الى الدالة وتقوم الدالة بطباعة عناصر المصفوفة كما يتضح فى نتيجة التنفيذ.

```
0:  /*Program Name CS6_9.C*/
1:  #include <stdio.h>
2:  void display(char array[5][10],int no);
3:  main()
4:     {
5:        char name [5] [10] = {
6:                             {" ahmed   "} ,
7:                             {"mohamed"} ,
8:                             {"samy     "} ,
9:                             {" hamdy   "} ,
10:                            {"nabil    "} ,
11:                                      };
12:       display(name,5);
13:    }

14:  void display(char array[5][10],int no)
15:    {
16:       int i;
17:       for (i=0;i<no;i++)
18:         {
19:           printf("\n %s",array[i]);
20:         }
21:    }
```

شكل رقم (٩-٦) ارسال المصفوفة الى الدالة

وعن هذا البرنامج نوضح ما يلى :

- يبدأ البرنامج فى السطر رقم ٥ بالاعلان مصفوفة حرفية واعطائهـا قيـم ابتدائيـة هى مجموعة اسماء
- يستدعى السطر رقم ١٢ الدالة display () و يرسل لها اسـم المصفوفـة وعـدد العناصر وفى السطر رقم ١٤ يبدأ انشاء الدالة display()

- ويقوم السطر رقم ١٩ بطباعة عناصر المصفوفة وذلك بتغيير رقم العنصر بالمتغير i حتى نحصل على نتيجة التنفيذ الموجودة فى شكل ١٠-٦ :

```
ahmed
mohamed
samy
hamdy
nabil
```

شكل ١٠-٦ طباعة عناصر المصفوفة

## دوال العبارات الحرفية string functions

هناك مجموعة كبيرة من الدوال التى تتعامل مع العبارات الحرفية وتستخدم فى كثير من الاحيان مثل المقارنة بين الكلمات و معرفة عدد حروف عبارة حرفية وتحويل كلمة من الحروف الصغيرة ( small) الى الحروف الكبيرة و نسخ سلسلة حروف فى سلسة حروف أخرى أو أضافة سلسلة الى أخرى وهكذا

ومعظم هذه الدوال موجودة فى الملف string.h ولعلك تذكر أن هذا الملف وجميع الملفات المشابهة له تأتى مع برنامج المترجم الخاص بلغة C وتوضع فى الفهرس المسمى include.

واليك أسماء بعض هذه الدوال والغرض من استخدامها ووظيفة كلا منها :

(اما اذا أردت معرفة جميع الدوال فعليك بفتح الملف string.4 من أى محرر سطور)

| الدالة | وظيفتها |
|---|---|
| strcat() | أضافة سلسلة حرفية (كلمة) الى نهاية سلسلة حرفية أخرى (كلمة أخرى) |
| strchr() | ايجاد ترتيب موقع حرف معين داخل كلمة |
| strcmp() | مقارنة كلمتين (أو متغيرين من نوع حرفى) |

| الدالة | وظيفتها |
| --- | --- |
| strcpy() | نسخ محتويات متغير حرفى فى متغير حرفى آخر |
| strlen() | ايجاد عدد حروف سلسلة حرفية |
| strlwr() | تحويل كلمة من الحروف الكبيرة الى الحروف الصغيرة |
| strncmp() | مقارنة حروف كلمة مع حروف كلمة أخرى |
| strncpy() | نسخ حروف متغير حرفى فى متغير حرفى أخر |
| strrev() | عكس حروف متغير حرفى |
| strsett() | جعل حروف متغير كلها من حرف واحد |
| strupr() | تحويل كلمة من الحروف الصغيرة الى الحروف الكبيرة |

وهناك مجموعة دوال اخرى تستخدم للتحويل من متغير رقمى إلى حرفى والعكس وفيما يلى أهم هذه الدوال

| الدالة | وظيفتها |
| --- | --- |
| atof | تحول متغير من نوع حرفى الى متغير من نوع رقم حقيقى |
| atoi | تحول متغير من نوع حرفى الى متغير من نوع رقم صحيح حتى يصلح التعامل معه كرقم |
| atol | تحول متغير من نوع حرفى الى متغير من نوع رقم صحيح طويل |
| _atold | تحول متغير من نوع حرفى الى متغير من نوع رقم حقيقى مضاعف |
| _itoa | تحول متغير من نوع صحيح الى متغير من نوع حرفى |
| _ltoa | تحول متغير من نوع رقم طويل الى متغير نوع حرفى |
| strtod | تحول متغير من نوع حرفى الى متغير من نوع رقم حقيقى مضاعف |

## أوامر المترجم preprocessors

أوامر المترجم هي أوامر تنفذ في حالة ترجمة البرنامج فقط وبالتالي هي أوامر توجه الى المترجم. وعندما يقابل المترجم أحد هذه الأوامر يقوم بتنفيذ ما يدل عليه هذا الامر ويطلق على هذه الاوامر preprocessaors أو directive.

وكلمة directive معناها توجيه أى أمر الى المترجم أما preprocessor فمعناه قبل التنفيذ والكلمتان تحققان المعنى فكل أمر يوجه الى المترجم يحقق غرض معين وينفذ في حالة الترجمة ومن هذه الاوامر مايلي

| | | |
|---|---|---|
| #define | #error | #include |
| #elif | #if | #line |
| #else | #ifdef | #pragma |
| #endif | #ifndef | #undef |

وقد درسنا معا الامر define# في درس الماكروز وعرفنا أنه أومر يوجه الى المترجم فمثلا define a 5# تقول للمترجم ضع مكان كل حرف a القيمة ٥ وهكذا ودرسنا الجملة include# فمثلا <include<stdio.h># معناها عند ترجمة هذا البرنامج افتح الملف stdio.h وضمه الى الملف الحالي

أما if# و else# و endif# فهي تقوم بنفس العمل الذى تقوم الجملة if والجمله else داخل البرنامج ولكن مع المترجم ، فمثلا if# تقول للمترجم اذا كان الشرط صحيح قم بعمل كذا و else# تقول والا قم بعمل كذا و endif# تنهي الجملة الشرطية.

والأمر ifdef# معناها اذا تم تعريف كذا نفذ الجملة التالية ، والأمر ifndef# معناه اذا لم يتم تعريف ... نفذ الجمل التالية وهكذا ...

وهناك ثوابت معرفة للجهاز ايضا تؤدى معنىمعيناً منها مايلي:

__LINE__ , __DATE__ , __TIME__ , __TIMESTAMP__ , __FILE__

فمثلا :

__LINE__ تعيد رقم السطر فى البرنامج الذى تكتب عنده هذا الثابت وتستخدم لتحديد رقم سطر الخطأ.

__DATE__ تعيد التاريخ الحالى

__TIME__ تعيد الوقت الحالى

__FILE__ تعيد اسم الملف الحالى الذى يجرى ترجمته

والبرنامج الموجود فى الشكل رقم ٦-١١ عبارة عن برنامج يقوم باستعمال بعض هذه الاوامر والثوابت ليعطى النتيجة الموضحة بالشكل رقم ٦-١٢ :

```
0:  /*Program Name CS6_11.C*/
1:  #ifdef _WINDOWS
2:  #define STRING "DOING A WINDOWS PROGRAM!"
3:  #else
4:  #define STRING "NOT DOING A WINDOWS PROGRAM"
5:  #endif
6:  int main(void)
7:    {
8:      printf( "\n\n") ;
9:      printf( STRING );
10:     #ifdef _MSC_VER
11:        printf( "\n\nUsing a Microsoft compiler!" );
12:        printf( "\n   Your Compiler version is %s", _MSC_VER );
13:     #endif
14:     #ifdef __TURBOC__
15:        printf( "\n\nUsing the Turbo C compiler!" );
16:        printf( "\n   Your compiler version is %x", __TURBOC__ );
17:    #endif
18:    #ifdef __BORLANDC__
19:        printf( "\n\nUsing a Borland compiler!" );
20:    #endif
```

```
21:    return(0);
22:  }
```

شكل رقم (١١-٦)استعمال الأوامر والثوابت المعرفة للمترجم

وعن هذا البرنامج نوضح مايلى :

يشتمل البرنامج على بعض التوجيهات وبعض الثوابت ففى السطر رقم ١ جملة #if تختبر هل البرنامج الحالى هو برنامج نوافذ أم لا وذلك باختبار الثابت _WINDOWS ويضع النتيجة فى المتغير STRING ويطلب السطر رقم ٩ طباعة نتيجة المقارنة

وبالمثل السطر رقم ١٠ و ١٤ و ١٨ لاختبار اصدار مترجم لغة C المستخدم

وعند تنفيذ هذا البرنامج نحصل على النتيجة التالية :

```
NOT DOING A WINDOWS PROGRAM
Using the Turbo C compiler!
Your compiler version is 295
```

شكل رقم ١٢-٦

والبرنامج الموجود بالشكل رقم ١٣-٦ يقوم باستعمال ثوابت الوقت والتاريخ ورقم السطر كما يظهر من نتيجة التنفيذ :

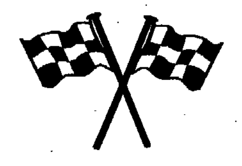
```
/*Program Name  CS6_13.C*/
#include <string.h>
int main(void)
  {
    printf("\n\nCurrently at line %d", __LINE__ );
    printf("\n\nThe value of __DATE__ is: ");
    printf(__DATE__);
    printf("\n\nThe value of __TIME__ is: ");
```

```
    printf(__TIME__);
    printf("\n\nThe value of __LINE__ is: %d", __LINE__ );
    printf("\n\nThe value of __FILE__ is: ");
    printf(__FILE__);
    return;
  }
```

شكل رقم ١٣-٦

وعند تنفيذ هذا البرنامج نحصل على النتيجة التالية :

```
Currently at line 14
The value of __DATE__ is: May 02 1995
The value of __TIME__ is: 22:56:18
The value of __LINE__ is: 22
The value of __FILE__ is: C:\CBOOK\CHP06\CS6_13.C
```



# الفصل السابع
# توظيف المفاتيح والتحكم فى حركة المؤشر

*فى هذا الفصل نتناول الموضوعات التالية:*

- *استخدام جدول الأكواد الممتدة* **EXTENDED CODE TABLE**
- *التعريف بملف* **ANSI.SYS** *واستخداماته ومتطلباته.*
- *التحكم فى حركة المؤشر*
- *وضع المؤشر فى أى مكان على الشاشة*
- *التحكم فى خصائص الحروف*
- *تصميم قوائم الاختيارات* **HIGH LIGHT MENU**

## توظيف المفاتيح

تتلخص فكرة توظيف أى مفتاح في استقبال حرف من لوحة المفاتيح واختبار هذا الحرف لتقرير هل هو الحرف المطلوب أم لا ،وبالتالى اتخاذ قرار بناء على نتيجة المقارنة ويكون الاختبار بطريقتين اما بمقارنة الحرف بالحرف أو بمقارنة كود الحرف المستقبل بكود الحرف المطلوب توظيفه .ونوضح فيما يلى مثال لكل حالة المثال الموجود بالشكل ١-٧ يختبر هل الحرف الذى يكتبه المستخدم هو حرف y فاذا كانت نتيجة المقارنة صحيحة يطبع الرسالة you typed y

```
main()
  {
    ch=getch();
    if(ch=='y')
    printf("\n you typed y ");
  }
```

شكل رقم ١-٧ برنامج لاختبار الحرف المدخل

المثال الموجود بالشكل ٢-٧ يختبر هل كود الحرف الذى يكتبه المستخدم يقابل حرف a أم لا ؟ فاذا كانت نتيجة المقارنة صحيحة يطبع الرسالة you typed a.

```
0:  /*CS7_2.C*/
1:  main()
2:    {
3:      int ch;
4:      ch=getch();
5:      if(ch==58)
```

```
6:      printf("\n you typed letter a ");
7:   }
```

شكل رقم ٢-٧ برنامج لاختبار كود الحرف المدخل

وفى هذا المثال نلاحظ أن :

السطر رقم ٤ يشتمل على الدالة () getch وهذه الدالة تستقبل حرف وتخزنه فى المتغير ch والسطر رقم ٦ يقارن اذا كان كود الحرف المستقبل هو الكود المطلوب أم لا فاذا كان هو يطبع الرسالة

هذا بالنسبة للمفاتيح العادية أى جميع الحروف التى لها كود موجود جدول الاكواد المعروف بـ ASCII TABLE والموجود بالملحق رقم (أ)

ولكن ماذا عن المفاتيح التى ليس لها كود فى هذا الجدول مثل :

F1,F2,ALT,CTR,..........

هذا ما ستعرفه فى البند التالى :

## توظيف مفاتيح الوظائف ومفاتيح التحكم

تشتمل لوحة المفاتيح بالاضافة الى المفاتيح العادية التى تستخدم فى الكتابة على مفاتيح أخرى يتم توظيفها غالبا وتختلف وظيفتها من برنامج لاخر ومن أمثلتها مفاتيح الوظائف F1 الى F12 أو مفاتيح التحكم مثل المفتاح Ctrl أو مفتاح Alt.

عند الضغط على مفتاح من المفاتيح العادية مثل .A,B,C ترسل لوحة المفاتيح للجهاز بايت واحدة ويتم اختبار هذه البايت وبالتالى توظيف المفتاح

ولكن عند الضغط على أحد مفاتيح الوظائف أو التحكم مثل المفتاح F1 ترسل لوحة المفاتيح ٢ بايت للجهاز

البايت الأولى لابد ان تكون صفر (٠) والبايت الثانية هى الكود الحقيقى للمفتاح. البايت الأولى هى التى تقول أن هذا المفتاح مفتاح ممتد أى من مفاتيح الوظائف وبالتالى ليس له ASCII CODE.

وبالتالى جميع المفاتيح الممتده مثل F1,F2,...ALT,CTR تشترك فى أن البايت الاولى لها صفر (٠) والبايت الثانية هى التى تميز كل مفتاح عن الاخر

وهناك جدول أخرلمثل هذه المفاتيح يسمى جدول المفاتيح الممتدة EXTENDED CODES TABLE.

ويوضح الجدول رقم ١-٧ الاكواد المخصصة للمفاتيح التى تأخذ ضغطة واحدة مثل............F1,F2,F3,

ويوضح الجدول رقم ٢-٧ أكواد المفاتيح التى تأخذ ضغطتين أو أكثر مثل ALT+C,CTR+A,.

| المفتاح | البايت الاولى | البايت الثانى |
|---|---|---|
| F1 | 0 | 59 |
| F2 | 0 | 60 |
| F3 | 0 | 61 |
| F4 | 0 | 62 |
| F5 | 0 | 63 |
| F6 | 0 | 64 |
| F7 | 0 | 65 |
| F8 | 0 | 66 |
| F9 | 0 | 67 |
| F10 | 0 | 68 |

| المفتاح | البايت الاولى | البايت الثانى |
|---|---|---|
| Home | 0 | 71 |
| Up arrow | 0 | 72 |
| PgUp | 0 | 73 |
| Left arrow | 0 | 75 |
| Right Arrow | 0 | 77 |
| End | 0 | 79 |
| Down | 0 | 80 |
| Pg Dn | 0 | 81 |
| Ins | 0 | 82 |
| Del | 0 | 83 |

جدول رقم ١-٧ أكواد المفاتيح المفردة

| المفتاح | البايت الاولى | البايت الثانى |
|---|---|---|
| Shift Tab | 0 | 15 |
| Alt Q,W,E,R,T,Y,U,I,O,P | 0 | 16 to 25 |
| Alt A,S,D,F,G,H,J,K,L | 0 | 30 to 38 |
| Alt Z,XC,V,B,N,M | 0 | 44 to 50 |
| Shift F1 to F10 | 0 | 84 to 93 |
| Ctrl F1 to F10 | 0 | 94 to 103 |
| Alt F1 to F10 | 0 | 104 to 113 |
| Ctrl PrtSc | 0 | 114 |
| Ctrl Left arrow | 0 | 115 |
| Ctrl right arrow | 0 | 116 |
| Ctrl End | 0 | 117 |
| PgDn | 0 | 118 |

| المفتاح | البايت الاولى | البايت الثانى |
|---|---|---|
| Ctrl Home | 0 | 119 |
| Alt 1,2,3,4,5,6,7,8,9,0, | 0 | 120 to 131 |
| Ctrl PgUp | | 132 |

جدول رقم ٢-٧ أكواد المفاتيح المشتركة

## مثال تحديد المفتاح وكوده

البرنامج الموجود بالشكل رقم ٣-٧ يختبر المفتاح الذى يضغطه المستخدم فإذا كان مفتاح عادى وكوده موجود بجدول الاكواد العادى يطبع رسالة تدل على ذلك بالإضافة إلى الكود الخاص به، واذا كان المفتاح من مفاتيح الوظائف يطبع رسالة تدل على ذلك بالإضافة الى كود المفتاح كما يظهر ذلك فى نتيجة التنفيذ فى الشكل رقم ٤-٧.

```
0:   /*CS7-3.C*/
1:   #include <stdio.h>
2:   main()
3:     {
4:       int key1,key2;
5:       while((key1=getch())!='\r')
6:         {
7:           if(key1==0)
8:             {
9:               key2=getch();
10:              printf("\n This is Exteneded  Code =%d",key2);
11:            }
12:          else
```

```
13:            printf("\n This Normal Code =%d",key1);
14:        }
15:    }
```

شكل رقم ٣-٧ تحديد المفتاح وكوده

```
Extended Code = 59
Extended Code = 60
Extended Code = 61
Extended Code = 62
Normal Code = 121
Normal Code = 106
Normal Code = 106
Normal Code = 104
```

شكل رقم ٤-٧ نتيجة التنفيذ

وعن هذا البرنامج نوضح ما يلى :

- فى السطر رقم ٤ اعلان عن متغيرين من نوع صحيح
- و السطر رقم ٥ يستقبل حرف ويخزنه في المتغير key1 ويستمر فى استقبال الحروف ما لم يضغط المستخدم المفتاح Enter و السطر رقم ٧ يقارن اذا كان هذا الكود هو صفر(٠) أى مفتاح ممتد مثل F1,F2, يرسل ٢ بايت ال المخزن المؤقت (buffer) ويشتمل السطر رقم ٩ على الدالة () getch التى تأخذ من الـ buffer بايت واحدة وتختبرها فاذا كانت صفر تقوم بطباعة رسالة وكود المفتاح كما فيالسطر رقم ١٠ فاذا لم تكن هذه البايت صفر فهذا معناه أن الحرف حرف عادى وبالتالى يطبع رسالة تدل على ذلك بالاضافة الى كود المفتاح العادى.وذلك فى السطر رقم ١٣

ولمعرفة كيفية توظيف المفاتيح الممتدة لتنفيذ دالة معينة تابع البرنامج الموجود بالشكل رقم (٥-٧) وهو برنامج ينتظر حتى يضغط المستخدم على مفتاح فاذا ضغط المستخدم مفتاح الوظيفة F1 يطبع البرنامج رسالة بذلك ويظهر ذلك فى نتيجة التنفيذ فى شكل رقم ٦-٧

```
0:  /*CS7-5.C*/
1:  #include <stdio.h>
2:  #include <conio.h>
3:  main ()
4:   {
5:     char key,key2;
6:     while ((key=getch()) !='\r')
7:     if (key ==0)
8:       {
9:          key2=getch();
10:         switch (key2)
11:            {
12:             case 59:
13:                printf ("\n function 1");
14:                break;
15:             case 60:
16:                printf ("\n function 2");
17:                break;
18:            default :
19:                printf ("\n other extended code ");
20:             }
21:       }
22:    else
23:       printf ("\n normal code  %3d",key);
24:  }
```

شكل رقم ٥-٧ توظيف مفاتيح الوظائف

```
function 1
function 2
other extended code
function 2
other extended code
normal code  102
normal code  116
normal code   55
normal code  121
other extended code
```

شكل رقم ٦-٧ نتيجة التنفيذ

وعن هذا البرنامج نوضح ما يلى :

- السطر رقم ٦ يستقبل حرف ويضعه فى المتغير key باستخدام الدالة ()getch
- فى السطر رقم ٧ تختبر جملة if كود الحرف المدخل لتحدد المفتاح من نوع ممتد (مفاتيح الوظائف) أم لا
- ثم فى السطر رقم ٩ الدالة ()getch تستقبل الكود الثانى وتخزنه فى المتغيرkey2 والذى يمثل الكود الحقيقى للمفتاح الممتد
- وفى السطر رقم ١٢ case تختبر هل هذا الكود هو 59 أى كود المفتاح F1 أذا كان كذلك اطبع الجملة function 1 وهكذا السطر رقم ١٥.
- أما السطر رقم ٢٢ فيقول اذا لم يكن الكود ممتد أى لم يكن الكود الاول ،نفذ السطر رقم ٢٣ الذى يطبع عبارة أنه مفتاح عادى

## إستعمال جدول الاكواد الممتدة EXTENDED CODE TABLE

من الجدول رقم ١-٧ والجدول رقم ٢-٧ والبرنامج الموجود بالشكل رقم ٧-٥ يتضح أن فكرة توظيف أى مفتاح ممتد هى معرفة كود هذا المفتاح من الجدول الخاص به واختباره فمثلا لتوظيف المفتاح F1 ليؤدى وظيفة help نكتب دالة بالاسم help() بالشكل الاتى :

```
case 59:
   help();
   break ;
```

على ان يتم كتابه محتويات الدالة help() فى البرنامج

## استخدام ملف ANSI.SYS

ملف ANSI.SYS هو ملف من انتاج المعهد الامريكى لتوحيد القياسات ANSI ويأتى هذا الملف مع ملفات نظام التشغيل DOS وهو يوفر مجموعة من الوظائف يمكن استغلالها مع معظم اللغات وكذلك لغة C

والوظائف التى يوفرها ملف ANSI.SYS يمكن تحقيقها بثلاث طرق :

- دوال لغة C مباشرة
- ملف ANSI.SYS
- استخدام ROM BOIS

وسوف نشرح فيما بعد تفصيل كل طريقة ونبدأ الأن بملف ANSI.SYS والوظائف التى يوفرها وكيفية استعمالها.

### *متطلبات ملف ANSI.SYS*

لكى تستخدم ملف ANSI.SYS يجب تهيئة الجهاز بملف CONFIG.SYS وذلك بكتابة السطر التالى فى الملف CONFIG.SYS

**DEVICE=ANSI.SYS**

بشرط أن يكون ملف ANSI.SYS موجود على الفهرس الرئيسى للقرص الذى تبدأ فيه تشغيل الجهاز أو ان يشتمل أمر path على اسم الدليل الذى يوجد تحته هذا الملف.

ومن داخل البرنامج يتم استدعاء ملف ANSY.SYS لتأدية وظيفة معينة بالشكل الأتى :

**printf("\x1B[code");**

تأمل الكود \x1B] هذا الكود ثابت ومعناه أنك تنادى ملف ansi.sys

أما الوظيفة المطلوبة فيكتب كودها مكان كلمة code وسيتضح ذلك من الامثلة القادمة.

### *التحكم فى حركة المؤشر*

المقصود بالتحكم فى حركة المؤشر (Cursor) استخدام المؤشر للتحرك لآعلى أو لاسفل أو يمينا أو يسارا أو صفحة لأعلى pageup أو صفحة لأسفل page down أو بداية السطر home أو نهاية السطر end وذلك باستخدام أكواد ملف ansi.sys من خلال الدالة ()printf التى تأخذ الشكل العام التالى :

**printf("\x1B[movement code");**

حيث movement code هو كود يحدد أتجاه الحركة المطلوبة فمثلا :

فى السطرالسابق تجد الكود الثابت لاستخدام ansi.sys وهو \x1B[ بالاضافة للحرف B والحرف B هو كود التحرك لاسفل بمقدار صف واحد (بالنظام السداسى عشر)

وبالتالى كل ما يفعله هذا الكود هو تحريك المؤشرلأسفل بمقدار صف واحد من موضعه الحالى.

والبرنامج الموجود بالشكل رقم (٧-٧) يوضح كيفية استخدام أكواد التحكم لتحريك المؤشر.

```
0:  /*CS7_7.C*/
1:  #include <sdtio.h>
2:  #include <conio.h>
3:  main ()
4:    { clrscr();
5:      while (getche() !='.')
6:        printf ("\x1B[B");
7:    }
```

شكل رقم ٧-٧ إستخدام أكواد Ansi.sys فى تحريك المؤشر

وعن هذا البرنامج نوضح ما يلى :

يشتمل هذا البرنامج على الدالة printf() التى تقوم بتحريك المؤشر لاسفل بمقدار صف واحد كلما أدخل المستخدم حرفا من لوحة المفاتيح ويوضح الجدول رقم ٣-٧ بقية الاكواد الخاصة بتحرك المؤشر cursor والتى تحتاج اليها فى تحريك المؤشر فى اتجاهات اخرى

وعند تنفيذ هذا البرنامج ستحصل على النتيجة التالية :

```
a
 b
  c
   d
    e
     f
      g
       h
        i
```

شكل رقم ٨-٧ نتيجة التنفيذ

| الكود | التاثير |
| --- | --- |
| "[2J" | مسح الشاشة |
| "[K" | مسح الحروف حتى نهاية السطر |
| "[A" | حركة المؤشر لاعلى |
| "[B" | حركة المؤشر لاسفل |
| "[C" | حركة المؤشر لليمين |
| "[D" | حركة المؤشر للشمال |
| "[%d;%df" | وضع المؤشر فى الصف والعمود المحددين |
| "[s" | حفظ الموضع الحالى للمؤشر |
| "[u" | استرجاع الموضع السابق للمؤشر |
| "[%dA" | حركة المؤشر لاعلى عدد صفوف A |
| "[%dB" | حركة المؤشر لاسفل عدد من الاعمدة B |
| "[%dC" | حركة المؤشر لليمين عدد من الاعمدة C |
| "[%dD" | حركة المؤشر للشمال عدد من الاعمدة D |

جدول ٣-٧ الأكواد المختلفة لحركة المؤشر

## الرسم باستخدام مفاتيح الاسهم

البرنامج الموجود بالشكل رقم ٩-٧ برنامج شامل يوضح كيفية توظيف مفاتيح الاسهم واستخدام ملف ansi.sys فى تحريك المؤشر والحصول على الرسم الحر

```
0:   /*CS7_9.C*/
1:   #define C_LEFT "\x1B[D"
2:   #define C_RIGHT "\x1B[C"
3:   #define C_UPUP "\x1B[A"
4:   #define C_DOWN "\x1B[B"
5:   #define L_ARRO 75
6:   #define R_ARRO 77
7:   #define U_ARRO 72
8:   #define D_ARRO 80
9:   #define ACCROSS  205
10:  #define UPDOWN   186
11:  void so();
12:   main ()
13:        {
14:            int key,key2;
15:            clrscr();
16:            while ((key=getch()) == 0)
17:              {
18:               key2=getch();
19:                switch (key2)
20:                  {
21:                    case L_ARRO:
22:                         printf (C_LEFT); printf ("%c",ACCROSS);
23:                         so();
24:                          break;
25:                    case R_ARRO:
```

```
26:                    printf (C_RIGHT); printf ("%c",ACCROSS);
27:                    so();
28:                    break;
29:                 case U_ARRO:
30:                    printf (C_UPUP); printf ("%c",UPDOWN);
31:                    so();
32:                    break;
33:                 case D_ARRO:
34:                    printf (C_DOWN); printf ("%c",UPDOWN);
35:                    so();
36:                    break;
37:                }
38:              printf (C_LEFT);
39:            }
40:     }

41:  void so()
42:     {
43:        int i;
44:        for (i=0;i<350;i++)
45            {
46              sound(i*4);
47:             delay(4);
48:             nosoubd();
49:           }
50:     }
```

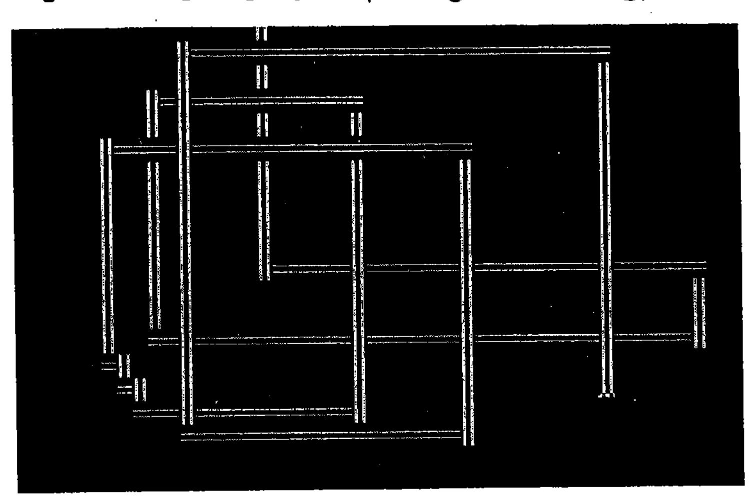
شكل رقم ٩-٧ برنامج الرسم الحر

وعن هذا البرنامج نوضح ما يلى :

تقوم فكرة هذا البرنامج على استقبال كود المفتاح وتحديد نوع المفتاح الذى تم الضغط عليه هل هو مفتاح الحركة لاعلى أو لاسفل أو اليمين أو الشمال و بعد

تحديد اتجاه الحركة يتم تحريك المؤشر الى الاتجاه المطلوب مع طباعة الحرف المستعمل فى الرسم وهكذا يستمر البرنامج بالسماح لك باستعمال مفاتيح الحركة فى الرسم حتى يتم الضغط على مفتاح الادخال.

وبتنفيذ هذا البرنامج والتحرك بمفاتيح الاسهم تحصل على شكل مشابه لشكل ١٠-٧.



شكل رقم ١٠-٧ نتيجة برنامج الرسم الحر

### *توجيه المؤشر الى أى مكان على الشاشة*

باستخدام ملف ansi.sys يمكنك وضع المؤشر فى أى مكان على الشاشة ويستخدم الصورة التالية :

```
printf("\x1B[R;Cf")
```

حيث R هى رقم الصف ، C هى رقم العمود ، وتكتب f كما هى لأنها تابعة للكود والبرنامج الموجود بالشكل رقم ١١-٧ يوضح ذلك حيث يقوم البرنامج باستقبال قيمتين صحيحتين ثم يقوم بتوجيه المؤشر الى هذا المكان على الشاشة مع طباعة العلامة * عند هذا المكان ليظهر موقعه على الشاشة.

```
0:  /*CS7_11.C*/
1:  #include <stdio.h>
2:  #include <conio.h>
3:  main ()
4:    {
5:       int x,y;
6:       while (1)
7:          {
8:             printf ("\n Enter Row,col:");
9:             scanf ("%d,%d",&x,&y);
10:            printf ("\x1B[%d;%df",x,y);
11:            printf ("%d,%d",x,y);
12:         }
13:   }
```

شكل رقم ١١-٧ برنامج توجيه المؤشر على الشاشة

## التحكم فى خصائص الحروف

من الاستخدامات المتاحة لملف Ansi.sys تغيير خصـائص الكتابـة والحـروف التى تظهر على الشاشة فمثلا يمكن عكس ألوان الكتابة بدلا من أن تكون باللون الابيـض على اللون الاسود يمكن عكسها لتظهر باللون الاسود على الابيض وكذلك يمكن الكتابة بحروف مائلة أومسطرة أو ثقيلة وهكذا.

ولتحقيق ذلك نستعمل الصورة التالية :

```
printf( "\x1B[NOm")
```

حيث يأخذ المتغير NO رقم يحدد الخاصية المطلوبة للحروف وهذا الرقم يتـم اختياره من الجدول الاتى :

| الرقم | تأثيره | مثال |
|---|---|---|
| 0 | Normal | أى الكتابة بخط عادى |
| 1 | Bold | أى الكتابة بخط ثقيل |
| 4 | Under Line | أى تسطير الكتابة |
| 5 | Blink | أى تجعل الكتابة تومض |
| 7 | Reverse | أى عكس الكتابة |
| 8 | Invisiple | أى إخفاء الكتابة |

والبرنامج الموجود بالشكل رقم ١٢-٧ يستخدم الاكواد الموجودة بالجدول السابق لطباعة مجموعة كلمات بأشكال مختلفة ويظهر ذلك فى نتيجة التنفيذ الموجودة بالشكل رقم ١٣-٧

```
0:   /*CS7_12.C*/
1:   #define NORMAL "\x1B[0m"
2:   #define BOLD  "\x1B[1m"
3:   #define UNDER  "\x1B[4m"
4:   #define BLINK  "\x1B[5m"
5:   #define REVERES "\x1B[7m"
6:   main ()
7:     {
8:    printf("NORMAL%s blinkng%sNORMAL\n\n",BLINK,NORMAL);
9:    printf("NORMAL%sBOLD%sNORMAL\n\n",BOLD,NORMAL);
10:   printf("NORMAL%sUNDER%sNORMAL\n\n",UNDER,NORMAL);
11:   printf("NORMAL%sREVERS%s NORMAL \n\n",REVERES,NORMAL);
12:   printf ("%s%s REVERSED AND BLINKING \n\n",REVERES,BLINK);
13:   }
```

الشكل رقم ١٢-٧ تغيير خصائص الحروف

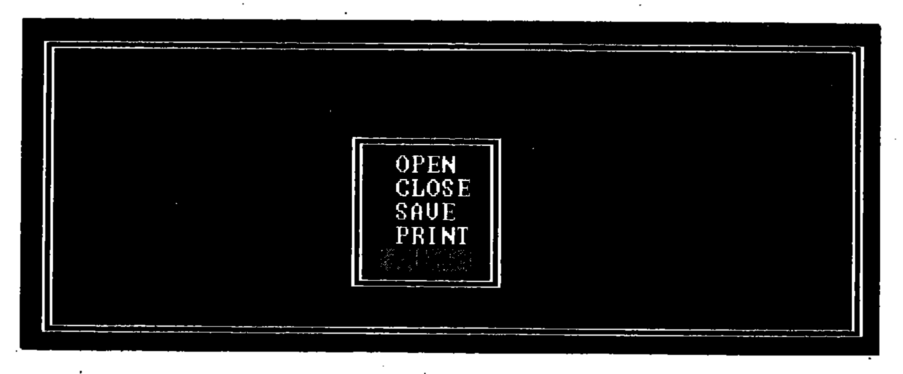
**وعند تنفيذ هذا البرنامج نحصل على النتيجة التالية :**

NORMAL **blinking** NORMAL
NORMAL **BOLD** NORMAL
NORMAL UNDER NORMAL
NORMAL REVERSE NORMAL
**REVERSED AND BLINKING**

الشكل رقم ١٣-٧ طباعة الخطوط باشكال مختلفة

## *برنامج لعمل قائمة اختيارات : HIGH LIGHT MENU*

**البرنامج الموجود بالشكل رقم ١٥-٧ يعتبر من التطبيقات الجذابة و هو عمل قائمة اختيارات ولكن بدلا من كتابة أرقام لاختيارات القائمة وكتابة رقم الاختيار يظهر شريط مضاء فوق أول اختيار ويقوم المستخدم بتحريك الشريط المضاء باستخدام مفاتيح الاسهم ثم الضغط على مفتاح الادخال عند الاختيار المطلوب (انظر شكل ١٤-٧).**




الشكل رقم ١٤-٧ استخدام الشريط المضاء فى قوائم الأختيارات

```
0:   /* Program Name CS7_15.C*/
1:   #define TRUE 1
2:   #define NUM 5
3:   #define CLEAR   "\x1B[2J"
4:   #define ERASE   "\x1B[K"
5:   #define NORMAL  "\x1B[0m"
6:   #define REVERSE "\x1B[7m"
7:   #define HOME    "\x1B[10;5f"
8:   #define BOTTOM "\x1B[20;1f"
9:   #define U_ARRO   72
10:  #define D_ARRO 80
11:  #define INSERT  82
12:  #include <stdio.h>
13:  #include <conio.h>
14:  #include <stdlib.h>
15:  void fancy_box(int x1,int y1,int x2,int y2)
16:    {
17:     int i;
18:     gotoxy(x1,y1);putch(201);
19:     for(i=x1+1;i<x2;i++) putch(205);
20:     putch(187);
21:     for(i=y1+1;i<y2;i++)
22:       {
23:         gotoxy(x1,i);putch(186);
24:         gotoxy(x2,i);putch(186);
25:       }
26:     gotoxy(x1,y2);putch(200);
27:     for (i=x1+1;i<x2;i++) putch(205);
28:        putch(188);
29:    }
30:  void action(int);
31:  void display (char *arr[],int size,int pos);
32:  char getcode(void);
```

```
33:  void main (void)
34:    {
35:       static char *items[NUM]= {
36:          " OPEN   ",
37:          " CLOSE ",
38:          " SAVE   ",
39:          " PRINT  ",
40:          " QUIT    " };
41:
42:       int curpos=0;
43:       textbackground(BLUE);
44:       textcolor(WHITE);
45:       clrscr();
46:       fancy_box(1,1,80,24);
47:       fancy_box(8,8,60,20);
48:       fancy_box(28,12,37,18);
49:       while (TRUE)
50:          {
51:             display(items,NUM,curpos);
52:             switch (getcode())
53:               {
54:                case U_ARRO:
55:                   if(curpos>0) --curpos;
56:                     else
57:                        curpos+=4;
58:                   break;
59:                case D_ARRO:
60:                   if(curpos<NUM-1) ++curpos;
61:                     else
62:                        curpos -=4;
63:                   break;
64:              case '\r':
65:                   action(curpos);break;
66:               }
```

```
67:          }
68:     }

69:  void display (char *arr[],int size,int pos)
70:     {
71:          int j;
72:          printf (HOME);
73:          for (j=0;j<size;j++)
74:             {
75:              if(j==pos)
77:                {
78:                 textcolor(RED);
79:                 textbackground(GREEN);
80:                }
81:                 gotoxy(15,10+j);
82:                 cprintf ("%s\r\n",*(arr+j));
83:                {
84:                 textcolor(WHITE);
85:                 textbackground(BLUE);
86:                }
87:             }
88:               printf (BOTTOM);
89:    }

90: char getcode(void)
91:    {
92:      int key;
93:      if ((key=getch() )==0)
94:           return (getch() );
95:        else if (key=='\r')
96:                 return(key);
97:              else
98:                 return (0);
99:      }
```

```
100:  void action(int pos)
101:      {
102:        printf (ERASE);
103:        switch (pos)
104:          {
105:            case 0:
106:               printf ("open\a");break;
107:            case 1:
108:               printf ("close\a");break;
109:            case 2:
110:               printf ("save\a");break;
111:            case 3:
112:               printf ("print\a");break;
113:            case 4:
114:               exit (0);
115:          }
116:      }
```

الشكل رقم ١٥-٧ برنامج القائمة ذات الشريط المضاء

وعن هذا البرنامج نوضح ما يلى :

ينقسم البرنامج الموجود فى الشكل رقم ١٤-٧ الى الاجزاء الاتية :

- من السطر رقم ٣ الى السطر رقم ١١ استخدامنا كلمة define فى عمل مجموعة تعريفات يتم استخدامها فى البرنامج.
- من السطر رقم ١٥ الى ٢٨ أنشأنا دالة لرسم مستطيل
- من السطر رقم ٣٥ الى٤٠ مصفوفة عبارات حرفية تحتوى على إختيارات الشاشة الرئيسية.
- فى السطر رقم ٤٩ الدوارة while1(TRUE) وهى الدوارة الانهائية
- فى السطر رقم ٥١ استدعاء الدالة ( ) display التى تقوم باظهار قائمة

الاختيارات والموجودة في المصفوفة items وتم انشاء هذه الدالة ابتداء من السطر رقم ٦٩.

وأنشأنا في هذا البرنامج مجموعة دوال لتحقق الغرض وهي :

الدالة display () تقوم بعرض اختيارات القائمة وتحديد مكان الشريط المضاء.

الدالة getcode() وتقوم باستقبال مفتاح من المستخدم واعادة كود هذا المفتاح الى الدالة الرئيسية.

الدالة action() والتي تقوم بتنفيذ المستخدم.

ويأخذ تنفيذ البرنامج التسلسل الاتي :

- في السطر رقم ٥١ الدالة display () تقوم بعرض قائمة الاختيارات على الشاشة.
- في السطر رقم ٥٢ الدالة getcode() تقوم باستقبال مفتاح من المستخدم وتقوم الدوارة while باختبار هذا المفتاح
- في السطر رقم ٥٤ أول حالة من الحالات المحتملة للمفتاح وهي حالة ضغط المستخدم على مفتاح السهم العلوي وفي هذه الحالة يتم انقاص قيمة المتغير curpos بمقدار واحد اذا كانت قيمة المتغير أكبر من صفر والا أخذ المتغير القيمة ٤ حتى يذهب الشريط المضاء الى أخر القائمة ثم تنتهي هذه الحالة ويتم الرجوع الى الدوارة while التي تقوم باستدعاء الدالة display () مرة ولكن بقيمة مختلفة للمتغير curpos تعبر عن حركة الاسهم وهكذا باقي الحالات.



# الفصل الثامن

# مؤشرات العناوين

# POINTERS

- يتناول هذا الفصل الموضوعات التالية .
- معنى المؤشر pointer
- مزايا استخدام المؤشر pointer
- اعادة أكثر من قيمة من الدوال
- المؤشرات والمصفوفات
- تمرير مصفوفات الى الدوال باستعمال المؤشرات
- الحرفيات والمؤشرات pointers and string
- مصفوفة الحرفيات والمؤشرات

## معنى المؤشر Pointer

المؤشر (pointer) هو نوع من أنواع البيانات ويعرف بأنه متغير يحتفظ(يخزن به) بعنوان مكان في الذاكرة.

من المعلوم أن كل مكان في الذاكرة له عنوان والجهاز يتعامل مع هذا المكان بالعنوان المحدد له ونحن بطريقة غير مباشرة نتعامل مع هذا العنوان، فمثلا هذا الإعلان int a=5; معناه احجز مكان في الذاكرة (RAM) حجمه ٢ بايت (حجم int) واجعل اسمه a وضع فيه القيمة ٥

وبالتالي كلما تعاملنا مع المتغير a فنحن نتعامل مع القيمة المخزنة فيه وليس العنوان المخصص لهذه القيمة . هذا عن الإعلان العادى ، فماذا عن الإعلان عن المؤشر (Pointer) هذا ما سنوضحه في البند التالي

## الاعلان عن المؤشر pointer

يتم الاعلان عن مؤشر الى أى متغير من أنواع البيانات بنفس الطريقة التى نعلن بها عن البيانات العادية وهى تحديد نوع البيانات ثم اسم المتغير ولكن الفرق بين الاعلان عن المتغير والاعلان عن المؤشر أن اسم المتغير يجب أن يسبق بالعلامة * ليدل على أنه مؤشر ، أى أن العلامة * تجعل المتغير مؤشر. فمثلا للاعلان عن مؤشر من نوع صحيح نكتب الصورة التالية

```
int  * p;
```

وكما ترى ليس هناك جديد غير أن اسم المتغير سبق بالعلامة *

*وماذا يعنى هذا الاعلان ؟*

يعنى أن المتغير p أصبح مؤشر الى مساحة فى الذاكرة مقدارها ٢ بايت مع الإحتفاظ بعنوان هذا المكان فى المتغيرp

هل لاحظت كلمة عنوان هذا ما يهمنا، وكلما أردنا أن نتعامل مع هذه القيمة تعاملنا عن طريق العنوان أى بدلا من أن نتعامل نحن مع القيمة ونترك الجهاز يتعامل مع العنوان بهذا الاسلوب نستطيع أن نتعامل مباشرة مع عنوان المكان مما يعطينا القدرة على عمليات كثيرة منها التعامل مع مخارج الجهاز مثل مخرج آلة الطباعة حيث أن لمخرج الطابعة عنوان فنستطيع أخذ هذا العنوان وتخزينه فى متغير ثم التعامل مع هذا المتغير كما نشاء وكذلك الكتابة فى ذاكرة العرض مباشرة وهكذا.



يمكن للمؤشر أن يشير الى أى نوع من أنواع البيانات حسب الاعلان

شرحنا كيف يتم الاعلان عن مؤشر يشير الى قيمة صحيحة فكيف يكون الاعلان عن مؤشر يشير الى قيمة حقيقية (pointer to float)

يكون ذلك بالصورة التالية

```
float  *k;
```

ومعناه أحجز مكان فى الذاكرة مقداره ٤ بايت وخزن عنوان هذا المكان فى المتغير k الذى يحتفظ بهذا العنوان

اذن طريقة الاعلان عن مؤشر الى أى نوع من أنواع البيانات هى نفس الطريقة المستخدمة للاعلان عن المتغيرات غير أننا نسبق المتغير بالعلامة * وهذا يعنى أنه مؤشر الى هذا النوع

## مزايا استخدام المؤشرات pointer

*يحقق استعمال المؤشرات فوائد كثيرة منها*

- اعادة أكثر من قيمة من الدوال.
- التعامل مع المصفوفات والحرفيات وتمريرها الى الدوال بشكل أفضل
- انشاء أنواع أكثر قوة من البيانات
- التعامل مع الجهاز ومكوناته وعناوين مداخل ومخارج الجهاز

## اعادة أكثر من قيمة من الدوال

من الفوائد المشهورة للمؤشرات استخدامها فى اعادة أكثر من قيمة من الدالـة. فما معنى ذلك ؟

فى الفصل الخامس شرحنا الدوال والتعامل معها وكيفية اعـادة قيمـة مـن الدالـة لاحظنا فى الأمثلة التى استخدمناها أننا استخدمنا كلمة return مرة واحدة مـع كـل دالـة وهذا معناه عدم امكانية اعادة أكثر من قيمة من الدالة.

فلو فرضنا أن لدينا مجموعة عمليات وأردنا انشاء دالة لهـذه العمليـات وانشـأنا الدالة وتم حساب نتائج العمليات ووضعت هذه النتائج فـى متغيرات وأردنـا أعـادة هـذه القيم الى الدالة الرئيسية ، هنا تظهر المشكلة. وهـى أننـا لا نسـتطيع اسـتعمال أكـثر مـن كلمة return وكلمة return لا تعيد الا قيمة واحدة أما فى حالة استخدام المؤشـرات فيمكننا إعادة أكثر من قيمة.

ولتوضيح ذلك سـنكتب برنامجـان البرنـامج الأول بـدون اسـتعمال المؤشـرات وفيه ستظهر هذه المشكلة ، والبرنامج الثانى باستخدام المؤشـرات ومنـه سـتعرف كيـف يمكن حل هذه المشكلة

يشتمل الشكل رقم ١-٨ على برنامج يقوم بانشاء دالة تأخذ معـاملين مـن نـوع صحيح ثم تقوم الدالة باضافة القيمة ٥ الى كل معامل

```
0:  /* Program Name CS8_1.C*/
1:  #include <stdio.h>
2:  #incldue <conio.h>
3:  void get2(int xx, int yy);
4:  main ()
5:    {
6:        int x=4,y=7;
7:        get2(x,y);
8:        printf("first no:is %d secand no:%d",x,y);
9:     }
10:  void get2(int xx,int yy)
11:   {
12:        xx+=5;
13:        yy+=5;
14:    }
```

الشكل رقم ١- ٨انشاء دالة داخل البرنامج

وعن هذا البرنامج نوضح مايلى :

يبدأ البرنامج الموجود فى الشكل رقم ١-٨ فى السـطر رقـم ٣ بـالاعلان عـن الدالة () get2 التى تأخذ معاملان من نوع صحيح وفي السطر رقم ٧ يتم استدعاء الدالة مع ارسـال قيمتيـن صحيحتيـن لهـا تأخذهمـا الدالـة الـى السـطر رقـم ١٠ وتضعهمـا فـى المتغيرين xx,yy.

وفي السطرين رقم ١٢و١٣ يتم اضافة القيمة ٥ الى كل من المتغيرين ثـم تقـوم الدالة الرئيسية فى السطر رقم ٨ بطباعة قيم المتغيرين.

*والسؤال هنا ما هي القيم التي يطبعها البرنامج ؟* هل يطبع البرنامج القيم بعد اضافة القيمة ٥ الى كل متغير كما في سطور الدالة أما يطبعها كما هي ؟.قبل أن تتابع شرح البرنامج حاول الوقوف والتفكير في ذلك.

قد تظن للوهلة الأولى أن النتيجة هي ٩ و ١٢ وذلك بعد اضافة القيمة ٥ الى القيمتين ٤ ، ٧ في جين أنك لو دققت النظر ستجد أن الدالة من النوع void كما في الاعلان في السطر رقم ٣ وبالتالي الدالة لاتعيد قيم وهذا ماتم حيث قامت الدالة باستقبال القيم واضافة القيمة ٥ الى كل عنصر ، ولكن لم تعيد الدالة النتيجة وبالتالي تظل قيم المتغيرات كما هي ٤و٧ وحتى لو أعلنا أن نوع الدالة int فلن تستطيع الدالة اعادة أكثر من قيمة. اذا كيف نعيد قيم المتغيرين في البرنامج السابق بعد اضافة القيمة ٥ اليهما هذا ما نراه من خلال البرنامج الموجود في الشكل رقم ٢-٨.

```
0:  /*Program Name CS8_2.C*/
1:  #include <stdio.h>
2:  #include <conio.h>
3:  void rets(int *xx,int *yy);
4:  main ()
5:    {
6:       int *x=5,*y=10;
7:       rets(&x,&y);
8:       printf ("first no:is %d secend is %d",x,y);
9:    }
10: void rets (int *xx,int *yy)
11:   {
12:      *xx+=5;
13:      *yy+=10;
14:   }
```

شكل رقم ٢-٨ استخدام المؤشرات مع الدوال

وعند تنفيذ البرنامج تحصل على النتيجة التالية

first no:is10 secand is 20

وعن هذا البرنامج نوضح ما يلي :

فى السطر رقم ٣ اعلان عن دالة لها معاملان وهذان المعاملان من نوع مؤشر الى قيمة صحيحة ، ومعنى أن المعاملات من نوع مؤشرات أننا فى حالة استدعاء الدالة لن نرسل الى الدالة قيم ولكن نرسل الى الدالة عناوين هذه القيم ، وهذا ما تم فى السطر رقم ٧ حيث تم ارسال عناوين المتغيران x ,y وذلك بالصورة &x,&y فوجود العلامة & مع المتغير يجعل المتغير يشير الى عنوان المكان وليس القيمة المخزنة فى المتغير ففى هذا السطر يتم ارسال عناوين المتغيرين x,y الى الدالة rets() التى تقوم باستقبال العناوين والتعويض بهما فى المتغيرين xx,yy ،وتقوم الدالة بزيادة القيم الموجودة فى هذه العناوين. أى أن ارسال العناوين يجعل الدالة تتعامل مع القيم الموجودة فى هذه العناوين وبالتالى تكون نتيجة هذا البرنامج هى طباعة القيم بعد زيادة القيمة المخزنة فى المتغير. وبهذا الاسلوب كأننا اعادنا قيمتين من الدالة. هل لاحظت فكرة ارسال عنوان الى دالة اذا ما فائدة هذه الفكرة ؟

الفائدة هى امكانية انشاء دالة تقوم بعمليات كثيرة وتخرج أكثر من ناتج أما بدون استعمال المؤشرات فلا تستطيع ان تعيد هذه القيم الى الدالة الرئيسية لانك لا تستطيع استعمال أكثر من جملة return.

ولكن يمكن كما فى هذا المثال أن نستدعى الدالة بحيث نرسل لها عناوين أى عدد من المتغيرات والدالة بدورها تجرى العمليات المطلوبة ثم تضع الناتج فى هذه العناوين ونستخدمها نحن من داخل الدالة الرئيسية.

للاعلان عن مؤشر تضع العلامه * قبل المتغير وللتعامل مع عنوان المكان مع الدالة يسبق المتغير بالعلامه & فمثلاً نكتب p* للاعلان عن مؤشر حسب النوع

ونكتب &p لإرسال عنوان المكان

## المؤشرات والمصفوفات Pointers and Arrays

للمؤشرات دور مع المصفوفات حيث تتعامل مع عناوين عناصر المصفوفات ، وهذا بالطبع أسرع من الطريقة المعتادة. وقبل أن نوضح كيف تتعامل المؤشرات مع المصفوفات نورد مثالآ يذكرنا بالتعامل مع المصفوفات بدون استعمال المؤشرات. والبرنامج الموجود بالشكل رقم ٣-٨ يقوم بالاعلان عن مصفوفة واعطائها قيم ابتدائية ثم يقوم بطباعة هذه القيم على الشاشة.

```
0:  /*Program Name CS8_3.C*/
1:  #include <stdio.h>
2:  #include <conio.h>
3:    main ()
4:     {
5:        int nums[ ]={92,81,70,69,58};
6:        int dex;
7:        for (dex=0;dex<5;dex++)
8:            printf ("\t%t",nums[dex]);
9:     }
```

شكل رقم ٣-٨ التعامل مع المصفوفة بدون مؤشرات

وعند تنفيذ هذا البرنامج نحصل على النتيجة التالية

```
92     81     70     69     58
```

اما فى حالة استخدام المؤشرات مع المصفوفات فيتم التعامل مع عناصر المصفوفة عن طريق اسم المصفوفة ، حيث يعتبر اسم المصفوفة هو عنوان أول عنصر فيها فمثلا المصفوفة int a [10] يعتبر اسم المصفوفة وهو a هو عنوان أول عنصر. فاذا طبعنا قيمة المتغير a نحصل على عنوان أول عنصر فى المصفوفة ولطباعة قيمة أول عنصر نسبق اسم المصفوفة بالعلامة * وبالتالى الصورة a* تعبر عن قيمة أول عنصر واذا

أضفنا ١ الى عنوان المصفوفة ستحصل على عنوان ثانى عنصر فمثلاً (a+1)* تشير الى قيمة ثانى عنصر وهكذا. والبرنامج الموجود بالشكل رقم (٤-٨) يوضح ذلك

```
0:  /*Program Name CS8_4.C*/
1:  #include <stdio.h>
2:  #include <conio.h>
3:   main ()
4:    {
5:          int nums[ ]={92,81,70,69,58};
6:          int dex;
7:          for (dex=0;dex<5;dex++)
8:              printf ("\t%d",*(nums+dex));
9:   }
```

الشكل رقم ٤-٨ استخدام المؤشرات مع المصفوفات

وعن هذا البرنامج نوضح ما يلى :

- فى السطر رقم ٥ اسم المصفوفة هو nums وهو فى نفس الوقت عنوان أول عنصر كما اشرنا بمعنى لو كتبت السطر ;printf("%d",nums) سوف يطبع لك عنوان أول عنصر تم حجزه لهذه المصفوفة حيث أنه فى حالة الاعلان عن مصفوفة يتم حجز أماكن بعدد العناصر ولكن يشار الى أول عنصر فقط باسم المصفوفة وتميز نهاية المصفوفة العلامة '\0' فإذا وضعنا العلامة * بجانب numsبالشكل nums* ؟ فإننا نشير الى القيمة الموجودة بهذا العنوان وبالتالى الصورة ;printf ("%d",*nums) تطبع قيمة أول عنصر.
- والصورة ;("%d",nums+1) تطبع عنوان العنصر الثانى
- والصورة ;("%d",*(nums+1)) تطبع قيمة العنصر الثانى
- والصورة ;printf ("%d",nums+2) تطبع عنوان العنصر الثالث
- والصورة ;printf ("%d",*(nums+2)) تطبع قيمة العنصر الثالث وهكذا

وهذا ما يتم فى البرنامج السابق حيث يتم زيادة العنوان بمقدار واحد بزيادة قيمة المتغير dex فى السطر رقم ٨

وبالتالى طباعة عناصر المصفوفة وبهذه الطريقة تتعامل المؤشرات مع المصفوفات

وعند تنفيذ هذا البرنامج نحصل على النتيجة التالية

```
92      81      70      69      58
```

## ارسال المصفوفة الى الدالة كمعامل

شرحنا فى درس الدوال أن معامل الدالة الذى نرسله لها عند الاستدعاء يمكن أن يكون متغير صحيح أو متغير حقيقى و يمكن أيضا أن يكون عنوان مكان فكيف تكون المصفوفة معامل للدالة ؟

يتم ذلك بارسال اسم المصفوفة الذى هو عنوانها وعدد العناصر إلى الدالة وبالتالى تأخذ الدالة الاسم (العنوان) وعدد العناصر وتتعامل مع هذه العناصر حسب العمليات الموجودة بالدالة. والبرنامج الموجود بالشكل رقم ٥-٨ يوضح كيفية استعمال المصفوفة كمعامل للدالة

```
0:  /*Program Name CS8_5.C*/
1:  #define SIZE 5
2:  void  addcon(int *ptr,int num,int con);
3:  main ()
4:    {
5:       int array[SIZE]={3,5,7,9,11};
6:       int konst=10;
7:       int j;
8:       addcon(array,SIZE,konst);
9:       for (j=0;j<SIZE;j++)
10:          printf ("%d\t",*(array+j));
11:  }
12: void  addcon(int *ptr,int num,int con);
```

```
13:    {
14:      int k;
15:      for (k=0;k<num;k++)
16:        *ptr=*(ptr++)+con;
17:    }
```

شكل رقم ٥-٨ استعمال المصفوفة كمعامل للدالة

وعن هذا البرنامج نوضح ما يلى :

- فى السطر رقم ٢ اعلان عن دالة لها ثلاث معاملات الأول مؤشر (pointer) والثانى والثالث رقم صحيح
- فى السطر رقم ٥ اعلان عن مصفوفة واعطائها قيم ابتدائية
- فى السطر رقم ٨ استدعاء للدالة addcon() وتأخذ ثلاث معاملات الاول اسم المصفوفة (عنوان المصفوفة) والثانى size و هو عدد عناصر المصفوفة والثالث رقم صحيح مقداره ١٠ ليتم جمعه على كل عنصر من عناصر المصفوفة
- فى السطر رقم ١٢ تبدأ الدالة وتستقبل المعاملات كما أشرنا اليها
- وفى السطر رقم ١٦ تتعامل الدالة مع عناوين العناصر فتجمع على كل قيمة داخل كل عنوان القيمة الثابتة المرسلة ١٠
- وفى الدالة الرئيسية فى السطر رقم ١٠ يتم طباعة عناصر المصفوفة بعد استدعاء الدالة.

وعند تنفيذ هذا البرنامج نحصل على النتيجة التالية :

```
13    15    17    19    21
```

## المؤشرات والعبارات الحرفية Pointers and strings

كما أن للمؤشرات (Pointers) دور مع المصفوفات فلها أيضا دور مع العبارات الحرفية. والحقيقة أن للمؤشرات دور مع جميع عناصر اللغة فلها دور مع الملفات(FILES) ولها دور مع دوال التعامل مع الذاكرة MEMORY ALLOCATION() وبالتالى لابد من فهم موضوع المؤشرات فهماً جيداً.تستخدم

المؤشرات مع العبارات الحرفية لزيادة سرعة تنفيذ البرنامج .وطريقة الاعلان المعتادة عن العبارات الحرفية بدون استخدام المؤشرات تأخذ الشكل التالى

```
char name[10];
```

وهو عبارة عن سلسلة من الحروف عدد ها ١٠ ولكن مع المؤشرات هناك طريقة أخرى للاعلان عن العبارة الحرفية مثل char *name="SAMY" هذا الاعلان يعنى أن المتغير name مؤشر يشيرالى عبارة حرفية ولكن ما طول هذه العبارة الحرفية ؟ يحدد ذلك من عدد حروف القيمة المعطاة ,وهذه الحالة عدد الحروف هو أربعة حروف وهو عد حروف كلمة SAMY.

والبرنامج الموجود بالشكل رقم ٦-٨ يوضح كيفية استعمال المؤشرات فى الاعلان عن عبارة حرفية

```
0:  /*Program Name CS8_6.C*/
1:  #include <stdio.h>
2:  #include <conio.h>
2:  main ( )
3:    {
4:       char *saluate="GREETING MR.";
5:       char name[7];
6:       puts("Enter your name:");
7:       gets(name);
8:       puts(saluate);
9:       puts(name);
10:   }
```

الشكل رقم ٦-٨ استعمال المؤشرات مع العبارة الحرفيه

وعن هذا البرنامج نوضح مايلى :

- فى السطر رقم ٤ يتم الاعلان عن مؤشر من نوع حرفى واعطائه قيمة ابتدائية هى الجملة GREETING MR وهذا شكل أخر للاعلان عن العبارة الحرفية

باستخدام المؤشرات وفى السطر رقم ٧ استقبال كلمة بالدالة gets() وتخزينها بالمتغير name

- وفى السطر رقم ٨ طباعة محتويات المتغير saluate وهو الجملة GREETING MR.
- فى السطر رقم ٩ طباعة محتويات المتغير name وهو SAMY بعد استقباله

وعند تنفيذ هذا البرنامج نحصل على النتيجة التالية :

```
Enter your name:
SAMY
GREETING MR.
SAMY
```

## مصفوفة المؤشرات Array of Pointer

كما يمكن استخدام المؤشرات في الاعلان عن عبارة حرفية ، يمكن أيضا استخدام المؤشرات فى الاعلان عن مصفوفة حرفيات والبرنامج الموجود بالشكل رقم ٨-٧ يقوم بالاعلان عن مصفوفة الحروف واعطائها مجموعة اسماء كقيم ابتدائية ثم يقوم البرنامج باستقبال اسم من المستخدم ويقارن هذا الاسم فى المصفوفة بمجموعة الاسماء المحزنة فى المصفوفة فاذا وجد هذا الاسم فى المصفوفة يطبع البرنامج رسالة بذلك واذا لم يجده يطبع رسالة اخرى.(وهو شكل أخر من البرنامج الموجود فى فصل المصفوفات)

```
0:  /* Program Name CS8_7.C*/
1:  #include  <stdio.h>
2:  #include  <conio.h>
3:  #define MAX 5
4:   main ( )
5:     {
6:        int d;
7:        int enter=0;
8:        char name[40];
```

```
9:      static char *list[MAX]=
10:             { "azab",
11:               "hamdy",
12:               "samy",
13:               "nabil",
14:               "mona" };
15:      clrscr();
16:      while(enter !=1) {
17:          printf ("\nEnter your name:");
18:          gets(name);
19:          for(d=0;d<MAX;d++)
20:          if(strcmp(list[d],name)==0)
21:             enter=1;
22             if (enter==1)
23:                 printf ("\nyour name is found..");
24:               else
25:                 printf("\nsorry your name not found.");
26:            }
27:      }
```

الشكل رقم ٧-٨

وعن هذا البرنامج نوضح ما يلى :

- فى السطر رقم ٩ اعلان عن مصفوفة عناصرها من نوع حرفيات (كلمات) مع اعطائها قيم ابتدائية هى مجموعة اسماء
- وفى السطر رقم ١٦ الدوارة while للاستمرار حتى شرط enter !=1
- وفى السطر رقم ١٨ الدالة gets() تستقبل مجموعة حروف عبارة عن اسم من المستخدم
- فى السطر ر قم ١٩ الدوارة for بعدد عناصر المصفوفة لتكرار عملية المقارنة بين الاسم المستقبل والاسماء المحزنة فى المصفوفة

- فى السطر رقم ٢٠ مقارنة بين كل عنصر من عناصر المصفوفة والاسم الذى أدخله المستخدم فاذا كان الاسم موجوداً ضمن الاسماء الموجودة فى المصفوفة تظهر الرسالة .. your name is found وينتهى البرنامج والا ظهرت الرسالة .sorry your name not found واستمر البرنامج فى التنفيذ وهكذا



فى البرنامج السابق فى السطر رقم 20 يمكن استبدال list [d] بالصورة (list+d) *

وهذا البرنامج يصلح ليصبح برنامج كلمة سر ولكن به مجموعة من الاسماء اذا كان المستخدم أحد هذه الاسماء سمح له البرنامج بالدخول فى العمل وأكمل له خطوات التشغيل والا لم يسمح له بالدخول فى البرنامج

وعند تنفيذ هذا البرنامج نحصل على النتيجة التالية :

```
Enter your name:ahmed
sorry your name not found.
Enter your name:mohamed
sorry your name not found.
Enter your name:samy
your name is found..
```



9

# الفصل التاسع

# السجلات STRUCTURES

فى هذا الفصل نشرح نوع من أنواع البيانات المهمة وهوالسجل (strucure) وهومفيد بالدرجة الاولى فى برامج قواعد البيانات.
يقابل كلمة Strucuer فى برامج قواعد البيانات كلمة Record
ويتناول هذا الفصل الموضوعات التالية :

- معنى السجل والحاجة الى استعماله
- انشاء السجل والاعلان متغير من السجل
- التعامل مع عناصر السجل (الحقول)
- استعمال السجل كعنصر فى سجل أخر
- مصفوفة السجلات Array Of Structure
- المؤشرات والسجلات
- تغيير نوع البيانات Type Casting
- اتحاد العناصر تحت اسم واحد union
- الفرق بين structure و union

## معنى السجل (STRUCTURE) والحاجة الى استعماله

من أهم التطبيقات فى عالم البرامج تطبيقات قواعد البيانات فمثلاً قاعدة بيانات موظفين تمثل بيانات الموظفين فى شكل سجلات كل سجل يتكون من مجموعة حقول ولو أن لك خبرة بأحد برامج قواعد البيانات مثل dbase فستعرف أن الملف ينقسم الى سجلات (records) والسجل ينقسم الى حقول (fields) ودائما نحتاج للتعامل مع السجل كوحدة وكذلك مع الحقول كوحدة. وتستخدم لغة C كلمة Stucture بنفس المفهوم الذى تستخدمه لغات البرمجة الأخرى لكلمة Record .

## استعمال السجل (STRUCTURE)

هناك خطوات تتبع للتعامل مع السجل وهى انشاء السجل (تركيب السجل) وتحديد الحقول المطلوبه ثم الاعلان عن متغير من نوع هذا السجل ثم التعامل مع حقول هذا السجل. والبرنامج الموجود بالشكل ١-٩ يشتمل على هذه الخطوات

```
0:  /*Program Name CS9_1.C  */
1:  #include <stdio.h>
2:  #include <conio.h>
3:  main ()
4:    {
5:       struct data
6:              {
7:               int num;
8:               char stat ;
9:               };
10:
11:      struct data stud;
12:      stud.num=5;
13:      stud.stat 't';
```

```
14:     printf ("\n stud.num=%d,stud.stat=%c",stud.num,stud.stat);
15: }
```

الشكل ١-٩ برنامج إنشاء السجل و إستخدامه

وعن هذا البرنامج نوضح مايلى :

- فىالسطر رقم ٥ يبدأ انشاء السجل وذلك باستعمال كلمة struct واعطاء هذا السجل اسم وهو data وكلمة data ممكن أن تكون أى كلمة.
- فى السطر رقم ٦ تبدأ مكونات هذا السجل بالقوس } وفى السطرين رقم ٧ ورقم ٨ اعلان عن حقول السجل وهىعبار عن متغير من نوع صحيح ومتغير من نوع حرف وينتهى السجل فى السطر رقم ٩ بالقوس {
- فى السطر رقم ١١ يتم الاعلان عن متغير من نوع السجل وهو المتغير stud وبالتالى أخذ المتغير stud نفس التركيب فأصبح له عنصر اسمه num من نوع صحيح وكذلك عنصر من نوع حرف وهو stat.
- فى السطرين رقم ١٢ ورقم ١٣ تم اعطاء قيم لحقول السجل ولكن الملاحظ أنه للتعامل مع حقل فى سجل يتم كتابة اسم الحقل مسبوقا باسم السجل التابع له وبينهما نقطة بالصورة stud.num
- فى السطر رقم ١٤ يتم طباعة قيم حقول السجل وبنفس الاسلوب تم كتابة اسم الحقل مسبوقا باسم السجل وبينهما النقطة للاشارة أن هذا الحقل تابع لهذا السجل.

وعند تنفيذ هذا البرنامج نحصل على النتيجة التالية :

```
stud.num=5,stud.stat=t
```



يمكن الاعلان عن أكثر من متغير من نوع السجل كما يحدث مع أنواع البيانات الاخرى فمثلا انظر الاعلان التالى struct data stud1,stud2,stud3 ومعناه أن المتغيرات stud1,stud2,stud3 من نوع data وبالتالى لها نفس العناصر.

وكذلك يمكن الاعلان عن متغيرات من نوع structure بطريقة أخرى نوضحها فيما يلى :

```
struct data
        {
        int num;
        char ch;
        }stud1,stud2;
```

ويعطى هذا الاعلان نفس النتيجة السابقة حيث أصبح stud1,stud2 لهما نفس تركيب السجل.

## كيفية ادخال بيانات الى عناصر السجل structure

يمكن معاملة عناصر السجل structure معاملة المتغيرات العادية حيث يمكن اعطائها قيم كما سبق ويمكن استقبال قيم بدوال الاستقبال من المستخدم ووضع هذة القيم فى عناصر السجل والبرنامج الموجود فى الشكل رقم ٢-٩ يقوم بانشاء سجل واستقبال قيم عناصره من المستخدم ثم طباعة هذه القيم على الشاشه

```
0:  /* Program Name CS9_2.C*/
1:  #include <stdio.h>
2:  main ()
3:     {
4:        struct data
5:            {
6:              int no;
7:              char name[10];
8:            };
9:       struct data stud;
10       printf("\n\n  stud.no=");
11:      scanf("%d",&stud.no);
12:      printf("\n\n  stud.name=");
13:      scanf("%s",stud.name);     /*Notes name is string*/
```

```
14:     clrscr();
15:     printf("\n\n stud.no=%d",stud.no);
16:     printf("\n\n  stud.name=%s",stud.name);
17:  }
```

شكل رقم ٢-٩ إنشاء السجل و إستقبال عناصره ثم طباعتها

وعن هذا البرنامج نوضح مايلى

- من السطر رقم ٤ الى السطر رقم ٨ تم انشاء السجل
- فى السطر رقم ٩ تم الاعلان عن متغير من نوع السجل
- فى السطر رقم ١١ استخدمنا دالة الاستقبال scanf() لاستقبال عناصر السجل وهنا يتضح الفرق بين التعامل مع متغير عادى ومتغير عنصر فى سجل وهو أننا ننسب عنصر السجل الى السجل التابع له وذلك عن طريق النقطة. بالصورة stud.no ومعناها الاشارة الى العنصر no التابع للسجل المسمى stud ويتم التعامل مع عناصر السجل كما يتم التعامل مع المتغيرات العادية.

وعند تنفيذ هذا البرنامج نحصل على النتيجة التالية:

```
stud.no=1
stud.name=mohamed
stud.no=1
stud.name=mohamed
```

## وضع محتويات سجل فى آخر

درسنا من قبل امكانية مساواة متغيرين من نوع واحد وذلك لوضع قيمة المتغير الأول فى المتغير الثانى وذلك بالصورة التالية :

```
int a,b;
b=5;
a=b;
```

وهذا معناه وضع القيمة المخزنة فى المتغير b وهى القيمة ٥ فى المتغير a ويمكن تحقيق ذلك مع السجلات بحيث يمكن مساواة متغير من نوع سجل مع أخر وبالتالى يتم مساواة قيم جميع العناصر بين السجلين بشرط أن يكون السجلين من نفس النوع، والبرنامج الموجود بالشكل ٣-٩ يوضح ذلك

```
0:  /* Program Name CS9_3.C*/
1:  #include  <stdio.h>
2:  #include  <conio.h>
3:  main ()
4:     {
5:          struct data
6:             {
7:                  int num;
8:                  char na;
9:             };
10:
11:    struct data stud1,stud2;
12:    stud1.num=5;
13:    stud1.na='t';
14:    stud2=stud1;
15:printf\("\n stud1.num=%d,stud1.na=%s",stud1.num,stud.na);
16:printf("\n stud2.num=%d,stud2.na=%s",stud2.num,stud.na);
17   }
```

شكل ٣-٩ وضع محتويات سجل فى سجل آخر

وعن هذا البرنامج نوضح مايلى :

- فى السطر رقم ٥ تم انشاء سجل جديد
- فى السطر رقم ١١ تم تعريف متغيرين من نوع السجل هما stud1,stud2
- فى السطر رقم ١٤ تم مساواة المتغيرين stud1 و stud2 وبالتالى وضع نسخة من القيم الموجودة فى السجلstud1 فى السجل الثانى stud2.

وعن تنفيذ هذا البرنامج نحصل على النتيجة التالية :

```
stud1.num=5,stud1.na=t
stud2.num=5,stud2.na=t
```

## السجلات المتداخلة Nested Structures

شرحنا أن السجل هو مجموعة من العناصر أيا كان نوع هذه العناصر وبالتالى يمكن أن تكون العناصر أو بعضها سجلات وهذا ما يسمى بالسجلات المتداخلة والبرنامج الموجود بالشكل رقم ٤-٩ يوضح كيف يكون السجل عنصر فى سجل آخر وكيفية التعامل مع عناصر السجلات فى هذه الحالة.

```
0:  /* Program Name CS9_4.C*/
1:  #include <stdio.h>
2:  main ()
3:    {
4:      struct person
5:         {
6:            int no;
7:            char name[10];
8:         };
9:      struct  group
10:        {
11:           struct person peno1;
12:           struct person peno2;
13:           int code;
14:        };
15:     struct group group1;
16:     printf("\n\n  group1.peno1.no=");
17:     scanf("%d",&group1.peno1.no);
18:     printf("\n  group1.peno1.name=");
19:     scanf("%s",group1.peno1.name);
20:     printf("\n group1.code=");
21:     scanf("%d",& roup1.code);
```

```
22:     group1.peno2=group1.peno1;
23:     clrscr();
24:     printf("\n\n the data of groups:\n\t");
25:     printf("\n group1.peno1.no=%d",group1.peno1.no);
26:     printf("\n group1.peno1.name=%s",group1.peno1.name);
27:     printf("\n group1.code=%d",group1.code);
28:     printf("\n group1.peno2.no=%d",group1.peno2.no);
29:     printf("\n group1.peno2.name=%s",group1.peno1.name);
30:  }
```

شكل رقم ٤-٩ السجلات المتداخلة

وعن هذا البرنامج نوضح ما يلي :

في هذا البرنامج تم الاعلان عن سجل يمثل بيانات اشخاص ثم تم الاعلان عن سجل أخر والعنصر الثاني في هذا السجل هو سجل من نوع السجل الأول وأخذ الاسم peno1 ثم تم التعامل مع عناصر السجلات كما في سطور البرنامج كما يلي

- من السطر رقم ٤ الى السطر رقم ٨ تم انشاء السجل الأول وهو person وعناصره هي no و name ومن السطر رقم ٩ الي السطر رقم ١٤ تم انشاء السجل الثاني هو group وعناصره هي code ويمثل كود المجموعة والسطر رقم ١٥ اعلان عن متغير من نوع السجل الثاني
- من السطر رقم ١٦ الى السطر رقم ٢٢ يتم استقبال العناصر ومن السطر رقم ٢٤ الى السطر رقم ٢٩ يتم طباعة عناصر السجل مع ملاحظة أننا ننسب كل عنصر الى السجل التابع له

وعند تنفيذ هذا البرنامج نحصل على النتيجة التالية :

```
group1.peno1.no=1
group1.peno1.name=hamdy
group1.code=01
the data of group:
```

```
group1.peno1.no=1
group1.peno1.name=hamdy
group1.code=01
group1.peno2.no=1
group1.peno2.name=hamdy
group1.code=01
```

## السجلات والدوال

*وهنا نناقش*

- استعمال السجل كمعامل للدالـــة
- اعادة السجل من الدالـــة

درسنا فى فصل الدوال أن الدالة يمكن أن تستعمل معاملات من أى نوع من البيانات وبالتالى يمكن أن يكون هذا المعامل من نوع سجل

بمعنى انه يمكن أن نرسـل للدالة سجل كمعامل والدالة بدورها تقوم بأى عمليات على هذا السجل وبالمثل يمكن للدالة أن تقوم ببعض العمليات على السجل ثم تعيد سجل الى الدالة الرئيسية ويكون نوع الدالة من نوع هذا السجل والبرنامج الموحود بالشكل ٥-٩ يقوم بالاعلان عن دالتين الاولى تقوم باستقبال عناصر سجل من المستخدم وعند استدعائها تعيد هذا السجل الى الدالة الرئيسية والدالة الثانية تأخذ هذا السجل كمعامل وتقوم بطباعة بياناته على الشاشة

```
0:   /*Prgram Name CS9_5.C*/
1:   struct personal
2:        {
3:          char name[30];
4:          int numb;
5:        };
6:    struct personal addname(void);
6:    void display(struet personal  custm);
7:    main ()
```

```
8:   {
9:       struct personal custmer1;
10:      struct personal custmer2;
12:      custmer1=addname();
13:      custmer2=addname();
14:      display(custmer1);
15:      display(custmer2);
16:   }
17:    /* addname()  */
18:    struct personal addname()
19:        {
20:           char numstr[81];
21:           struct personal custmer;
22:           printf ("\n new custmer \n Enter name:");
23:           gets(custmer.name);
24:           printf ("\n Enter custmer no:");
25:           gets(numstr);
26:           custmer.numb=atoi(numstr);  /*convert string to
                                                   integer*/
27:           return custmer;
28:        }
29:    /* list() */
30:   void display(struct personal custm)
32:      {
33:         printf("\n Custmers:\n");
34:         printf ("Name:%s\n",custm.name);
35:         printf ("Number:%d \n",custm.numb);
36:      }
```

شكل ٥-٩ السجلات والدوال

وعن هذا البرنامج نوضح مايلى :

فى السطر رقم ٦ تم الاعلان عن الدوال ()addname و ()display. والدالـة addname() من نوع struct personal فماذا يعنى أنها من هذا النوع ، نعرف أن نوع الدالة يعتمد على نوع القيمة التى تعيدها الدالة ولو نظرت الى كلمة return فـى الدالـة addname() في السطر رقم ٢٧ تجد أنها تعيد structure من نوع struct

personal لذلك لابد أن يكون نوعها مـن نفـس نـوع القيمـة التـى تعيدهـا وهـو struct personal. والدالة display لا تعيد قيم بل تقوم بطباعة قيم عناصرالسـجل structure فقط لذلك كان نوعها void

وعند تنفيذ هذا البرنامج نحصل علىالنتيجة التالية :

```
new custmer
Enter name:samy
Enter custmer no:1
new custmer
Enter name:hamdy
Enter custmer no:2
Custmer:
Name:samy
Number:1
Custmer:
Name:hamdy
Number:2
```

## مصفوفة السجلات Arrags Of Structures

درسنا فى درس المصفوفات أن المصفوفة هى مجموعة مـن العنـاصر مـن نفس النوع هذا النوع يمكـن أن يكـون أى نـوع مـن البيانـات وبالتـالى يمكـن أن نعلـن عـن

مصفوفة عناصرها من نوع سجلات والبرنامج الموجود بالشكل ٦-٩يقوم بانشاء سجل ثم الاعلان مصفوفة من هذا السجل ثم استقبال قيم عناصر هذه المصفوفة ثم طباعتها

```
/* Prgram Name CS9_6.C*/
#include <stdio.h>
main ()
   {
    int i;
   struct personal
        {
        char name[30];
        int numb;
        };
   struct personal custmer[5];
   for(i=0;i<5;i++)
       {
          printf("\n custmer.no[%d]=",i);
          scanf("%d",& custmer[i].numb);
          printf("\n custmer.nname[%d]=",i);
          scanf("%d", custmer[i].name);
       }
   for(i=0;i<5;i++)
       {
          printf("\n %d \t %s", custmer[i].numb, custmer[i].name);
       }
```

شكل ٦-٩ إستخدام مصفوفة السجلات

## المؤشرات والسجلات Pointers and Structures

ذكرنا أن المؤشرات (pointers) تدخل فى كل عناصر اللغة وبالتالى لها دور مع السجلات (structures) وهنا نناقش كيفية الاعلان عن مؤشر من نوع structure (pointer to structure) وكيفية استعمال هذا المؤشر مع السجلات (structures).

الفرق بين اعلان مؤشر الى سجل واعلان متغيرمن نوع سجل هو أن المؤشر يسبق بالعلامة * ونوضح ذلك من خلال البرنامج الموجود بالشكل رقم ٧-٩

```
0:  /* Prgram Name CS9_7.C*/
1:  void main (void)
2:    {
3:      struct xx
4:        {
5:          int num1;
6;          char ch;
7:        };
8:     struct xx xx1;
9:     struct xx *ptr;
10:
11:     ptr=&xx1;
12:     ptr->num1=303;
13:     ptr->ch='q';
14:     printf ("ptr->num1=%d\n",ptr->num1);
15:     printf ("ptr->ch=%c",ptr->ch);
16:   }
```

شكل رقم ٧-٩ إستخدام المؤشرات مع السجلات

وعن هذا البرنامج نوضح ما يلى

فى السطر رقم ٣ بدأ البرنامج بانشاء سجل ، وفى السطر رقم ٨ تم الاعلان عن متغير من نوع هذا السجل ، وفى السطر رقم ٩ تم الاعلان عن مؤشر الى هذا السجل وفى السطر رقم ١١ تم تخزين عنوان السجل xx1 فى المتغير ptr وبعد ذلك يتم التعامل مع السجل عن طريق التعامل مع المؤشر الى هذا السجل كما فى السطر رقم ١٢ والسطر رقم ١٣ ونلاحظ أن التعامل مع عنوان السجل وعناصر السجل هو نفس التعامل مع متغير السجل وعناصره غير أننا نستبدل الصورة xx.no بالصورة ptr->no حيث أن ptr عبارة عن مؤشر إلى السجل

وعند تنفيذ هذا البرنامج نحصل على النتيجة التالية

```
ptr->num1=303
ptr->ch=q
```

## تغيير نوع البيانات TYPECASTING

من العمليات المفيدة فى لغة C وجود أكثر من طريقة لتحويل بيانات من نوع الى نوع آخر وتتم بطريقتين:

*الطريقة الاولى* : استخدام مجموعة دوال مثل الدالة ()atoi بمعنى ascii to int وهى تقوم بتحويل الارقام التى فى صورة حرفيات الى صورة أرقام للتعامل معها كأرقام والدالة ()atof بمعنى ascii To float والتى ذكرناها فى فصل المصفوفات.

*الطريقة الثانية* : هى ما يسمى Typecasting وهى كتابة النوع المطلوب التحويل اليه قبل المتغير المطلوب تغير نوعه. والبرنامج الموجود فى الشكل رقم ٨-٩ يوضح فكرة تغير النوع باستخدام هذه الطريقة

```
0:  /* Prgram Name CS9_8.C*/
1:  #include<stdio.h>
2:  main ( )
3:     {
4:         int a=66,b;
5:         char ch='t',g;
6:          b=(int) ch;
7:           g=(char) a;
8:           printf("b=%d",b);
9:           printf("\n g=%c",g);
10:     }
```

شكل رقم ٨-٩ برنامج تغيير نوع البيانات

وعند تنفيذ هذا البرنامج نحصل على النتيجة التالية:

```
b=116
g=B
```

وعن هذا البرنامج : نوضح ما يلى :

يبدأ البرنامج فى السطر رقم ٤ بالاعلان عن متغيرين من نوع صحيح هما a,b و السطر رقم ٥ اعلان عن متغيرين من نوع حرف وهما ch,g وفى السطر رقم ٦ كتابة كلمة int قبل المتغير ch يؤدى الى تحويل الحرف الذى خزن فى المتغير ch الى كوده وبالمثل السطر رقم ٧ وهذا ما يسمى typecasting وبالتالى نحصل على النتيجة السابقة.

## اتحاد البيانات تحت اسم واحد UNION

من أنواع البيانات المشتقة الموجودة فى لغة C نوع يسمى union وهو يشبه بدرجة كبيرة السجل (STRUCTURE) مع وجود فارق بينهما وقبل معرفة هذا الفارق نوضح شكل UNION من خلال البرنامج الموجود بالشكل رقم ٩-٩

```
0:   /* Prgram Name CS9_9.C*/
1:    void main (void)
2:      {
3:         union intflo
4:            {
6:               int intnum;
7:               float fltnum;
8:            }unex;
9:        printf ("sizeof (union inflo)=%d\n",sizeof(union intflo);
10:       unex.intnum=734;
11:       printf ("unex.intnum=%d\n",unex.intnum);
12:       unex.fltnum=54.4;
13:       printf ("unex.fltnum=%.2f\n",unex.fltnum);
14:     }
```

الشكل رقم ٩-٩ إستخدام union داخل البرنامج

وعن هذا البرنامج نوضح ما يلى :

فى البرنامج الموجود بالشكل رقم ٩-٩ فى السطر رقم ٣ تم انشاء union ولايختلف انشاءه عن انشاء strucure ويتم التعامل معه كما يتم التعامل مع structure كما يظهر ذلك من سطور البرنامج ولكن الفرق بين union و structure هو أن الـ structure يحجز مساحة فى الذاكرة مقدارها مجموع عناصره بينما الـ union يحجز مساحة مقدارها مساحة أكبر عنصر فلو تأملت البرنامج السابق تجد أنه من السطر رقم ٣ الى السطر رقم ٨ تم تعريف union وتركيبه عبارة عن متغير صحيح و متغير حقيقى، ومن المفترض أن تكون المساحة الكلية لهذا الـ union هى مجموع مساحة المتغير الصحيح والحقيقى. أى ٦ بايت هذا ما يحدث مع السجل ولكن مع union يتم حجز مساحة أكبر عنصر وهو فى هذه الحالة المتغير الحقيقى أى ٤ بايت

وعند تنفيذ هذا البرنامج نحصل على النتيجة التالية

```
sizeof (union inflo)=4
unex.intnum=734
unex.fltnum=54.4
```

## لماذا نستخدم UNION

يمكننا استعمال الـ union من استقبال أكثر من نوع من البيانات على متغير واحد ويظهر ذلك فى الشكل رقم ١٠-٩

```
union NO
   {
       double dno;
       float fno;
       long ino;
       int no;
```

```
    char cno;
    };
```

الشكل ١٠-٩ إستخدام union لإستقبال أكثر من نوع من البيانات



في حالة أستعمال union يقوم الـunion بحجز مساحة واحدة تستعمل لجميع عناصره وبالتالي كلما تم ادخال قيمه عنصر تلغى القيمه القديمه لأنها مساحة واحدة

في هذا المثال تم انشاء union عناصره double,float,long,int,char

وكما اتفقنا أن union يحجز في الذاكرة RAM مساحة مقدارها مساحة أكبر عنصر فقط وفي نفس الوقت يمكن استقبال جميع العناصر ويتم استقبالها على نفس المساحة وبالتالي تمكننا هذه الفكرة من استقبال عدة أنواع من البيانات وكأنه متغير واحد

## استعمال structure كعنصر من عناصر union

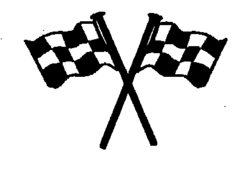
من أهم التطبيقات المشهورة لاستخدام الـ union انشاء union عناصره structures وسوف نشرح هذا المفهوم بالتفصيل في الفصل (ROM BIOS) ويسمى union of structures.

ويتضح ذلك في البرنامج الموجود بالشكل رقم ١١-٩

```
/* Prgram Name CS9_11.C*/
void main ()
  {
   struct twoints
          {
          int intnum1;
          int intnum2;
          };
```

```
union intflo
        {
        struct twoints setx;
        float fltnum;
        } unex;
printf ("sizeof (union intflo)=%d\n",sizeof(union intflo));
unex.setx.intnum1=723;
unex.setx.intnum2=-455;
  printf ("unex.setx.intunm1=%d\n",unex.setx.intnum1);
  printf ("unex.setx.intunm2=%d\n",unex.setx.intnum2);
  unex.fltnum=875.45;      /* we print structure element first */
  printf("unex.fltnum=%f\n",unex.fltnum);
}
```

شكل ١١-٩ إستعمال structure كعنصر من عناصر union



# الفصل العاشر

# الملفات

باستعمال الملفات يمكن حفظ البيانات بصفة دائمة، ولغة C تتميز بأن بها طرق كثيرة للتعامل مع الملفات نناقشها خلال الموضوعات التالية :

- الطرق المختلفة للتعامل مع الملفات
- الكتابة والقراءة حرف بحرف
- دوال فتح الملف وغلقه
- كتابة وقراءة عبا رة حرفية كل مرة
- التعامل مع الطابعة والملفات الثابتة
- كتابة وقراءة بيانات صحيحة وحقيقية وحرفية
- قراءة وكتابة سجل فى كل مرة
- قراءة وكتابة مجموعات من البيانات

ربما يكون التعامل مع الملفـات فـى لغـة C مختلفـاً شـيء مـا عـن التعـامل مـع الملفات فى لغات قواعد البيانات مثل dBASE وغيرها ، ولكن رغم سهولة التعـامل مـع الملفات فى اللغات الأخرى مثل dbase فهى محدودة بعمليـات معينـة تحصـر المبرمج فى اطار ملفات قواعد البيانـات ، أمـا لغـة C فتتميـز بـأن بهـا طـرق كثيـرة للتعـامل مـع الملفات تمكنك من التعامل مع أنواع محتلفه من الملفات. وفيما يلـى الطـرق المختلفـة للتعامل مع الملفات

## الكتابة حرف بحرف فى ملف (char by char)

من الطرق المتاحة فى لغـة c الكتابـة فـى ملـف حـرف بحـرف بمعنـى إسـتقبال حرف من المستخدم وتخزينه مباشرة فى ملف ،وقبل أن نبدأ فى شرح قواعد التعامل مــع الملف نتابع البرنامج الموجود فى الشكل رقم ١-١٠ الـذى يسـمح للمسـتخدم بكتابـة حرف بحرف مع تخزين هذا الحرف فى ملف ويستمر البرنامج فـى قبـول الحـرف حتـى يضغط المستخدم على مفتاح الادخال

```
0:  /*Program Name CS10_1.C*/
1:  #include <stdio.h>
2:  main ()
3:   {
4:      FILE *fptr;
5:      char ch;
6:      fptr=fopen("textfile.txt","w");
        printf("\n write to the file:\n");
7:      while ((ch=getche()) !='\r')
8:        {
9:          putc(ch,fptr);
10:       }
11:     fclose (fptr);
12:  }
```

شكل ١-١٠ الكتابة فى الملف حرف بحرف

وعن هذالبرنامج نوضح ما يلى :

- فى السطر رقم ٤ اعلان عن متغير من نوع مؤشر الى ملف ويستعمل هذا المؤشر للاشارة الى الملف الذى يتم فتحه مع ملاحظة.أن كلمة FILE تكتب بالحروف الكبيرة لانها معرفه بلغة C (الملف Stdio.h) ، وهى تجعل المتغير fptr من نوع مؤشر الى FILE أى يشير الى ملف.
- وفى السطر رقم ٦ يتم فتح ملف بالاسم textfile والامتداد txt للكتابة فيه وذلك باستعمال الدالة ()fopen وقبل أن نكمل شرح سطور البرنامج نشرح الدالة ( ) fopen والصورة العامة لدالة fopen هى.

```
fptr=fopen("filename.ext","mode");
```

- حيث المعامل filename هواسم الملف المراد فتحه
- والمعامل ext هو امتداد الملف الذى تريد فتحه وفى حالة انشاء ملف جديد فأن تحديد هذا الامتداد واختياره أمر اختيارى يرجع إليك ولذلك عند عمل برنامج متكامل يمكنك أن تحدد الامتداد الذى يروق لك
- والمعامل mode هو الحالة التى تريد فتح الملف من اجلها هل تريد فتح الملف للكتابة فقط ، أم للقراءة فقط ، أم للاضافة فقط ، أم للقراءة والكتابة ، يتم تحديد ذلك بكتابة الحرف الدال على الحالة المطلوبة ويوضح الجدول التالى الحرف التى تستخدم للحالات المختلفه :-

| الحرف | متى يستخدم |
|---|---|
| a | فى حالة فتح ملف للاضافة وبالتالى تبدأ الكتابة من نهاية الملف وما يكتبه المستخدم يضاف لمحتويات الملف الاصلى. واذا لم يكن الملف موجود يتم انشاءه والكتابة فيه |

| الحرف | متى يستخدم |
|---|---|
| "a+" | مثل الحالة السابقة ولكن تسمح هذه الحالة بالقراءة أيضا فيمكن لك إستخدام أوامر الكتابة وأوامر القراءة مع نفس الملف المفتوح.بهذه الحاله |
| "r" | تفتح الملف للقراءة فقط واذا لم يكن الملف المطلوب فتحه موجوداً على القرص أو إذا كان القرص غير صالح تعيد الدالة صفر أى لاتستطيع فتح الملف المذكور. |
| "r+" | تسمح بالقراءة والكتابة اى استعمال دوال القراءة والكتابة مع الملف ولكن الكتابة لاتعنى الإضافة فهى تكتب لأول مرة أو تكتب فوق البيانات الموجودة |
| "w" | تفتح ملف جديد للكتابة فقط فإذا كان الملف المطلوب فتحه موجوداً على القرص تكتب البيانات الجديدة فوقه ويتم الغاء محتوياته |
| "w+" | نفس الخالة السابقة ولكن تسمح بالقراءة والكتابة |

نعود الى شرح البرنامج

- في السطر رقم ٧ الدالة ( )getche تستقبل حرف وتخزنه فى المتغير ch والدوارة while تجعل الاستقبال يستمر ما لم يضغط المستخدم على مفتاح الادخال Enter.
- وفى السطر رقم ٩ الدالة ( ) putc تكتب الحرف الموجود فى المتغير ch فى الملف الذى يشير اليه المتغير fptr أى فى الملف textfile.txt وهكذا تظل الدالة getche() تستقبل حرف وفى نفس الوقت تقوم الدوارة while باختبار هذا الحرف وتقوم الدالة ( ) putc بكتابة هذا الحرف فى الملف، فاذا ضغط المستخدم على مفتاح الادخال enter ينتهى عمل الدوارة while وتقوم الدالة ( ) fclose بأغلاق الملف.

وعند تنفيذ هذا البرنامج نحصل على النتيجة التالية

```
write to the file :
allah the god of all world
```

## القراءة من ملف حرف بحرف (char by char)

في البرنامج السابق والموجود بالشكل رقم ١٠-١ شرحنا كيف تم فتح ملف للكتابة فيه وكيف نستقبل حروف من المستخدم ونكتبها في هذا الملف ، والبرنامج الموجود بالشكل رقم ١٠-٢ يوضح كيفية فتح ملف للقراءة بطريقة حرف بحرف ثم طباعته على الشاشة

```
0:  /*Program Name CS10_2.C*/
1:  #include <stdio.h>
2:  #include <stdlib.h>
3:  main (void)
4:    {
5:       FILE *fptr;
6:       char ch;
7:       clrscr();
8:       fptr=fopen("textfile.txt","r");
9:       printf("\n contents of the file:\n");
10:      while ((ch=getc(fptr)) !=EOF)
11:        {
12:          printf ("%c",ch);
13:        }
14:      fclose (fptr);
15:      getch();
16:      return 0;
17:  }
```

الشكل رقم ١٠-٢ قراءة ملف حرف بحرف

وعن هذا البرنامج نوضح ما يلي :

- في السطر رقم ٥ اعلان عن مؤشر الى ملف لاستعماله فى الاشارة الى ملف حيث تتم جميع العمليات مع الملف عن طريق هذا المؤشر
- وفى سطر رقم ٨ دالة فتح الملف وتلاحظ أن حالة الفتح حددت هنا بالحرف r أى فتح الملف للقراءة فقط.
- وفى سطر رقم ١٠ الدالة getc(fptr) تقرأ حرف من الملف المشار اليه بالمؤشر fptr وتضع هذا الحرف في المتغير ch وفى نفس الوقت تقوم الدوارة while بمقارنة هذا الحرف بالقيمة EOF وهو ثابت معرف بمعنى END OF FILE أى نهاية الملف وتستمرالدالة () getc في قراءة حرف من الملف ما لم يكن هذا الحرف هو القيمة EOF أى ما لم تصل الى نهاية الملف كله.
  وتقوم الدالة ( ) printf بطباعة هذا الحرف على الشاشة ويستمر ذلك حتى يتم عرض محتويات الملف.

وعند تنفيذ هذا البرنامج نحصل على النتيجة التالية :

```
contents of the file:
allah the god of all world
```

## المشاكل المتوقعة عند فتح ملف

هناك الكثير من المشاكل المتوقعة عند فتح ملف منها مايلى

- القرص المطلوب التعامل معه سواء للكتابة أو للقراءة غير صالح للاستعمال وبالتالى لاتستطيع الدالة ( ) fopen فتح الملف
- الملف غير موجودعلى الاسطوانة الحالية

بالتالى على المبرمج التأكد من أن عملية فتح الملف تمت بدون مشاكل ليتجنب ظهور رسالة خطأ من نظام التشغيل DOS وإنتهاء البرنامج بشكل غير مقبول

البرنامج الموجود بالشكل رقم ٣-١٠ يقوم بالتأكد أولا من فتح الملف فإذا لم يتم فتحه يعطى البرنامج رسالة خطأ وينتهى عمل البرنامج ، واذا تم فتح الملف يكمل البرنامج باقى الأوامر. فيفتح الملف ويعرض محتوياته

```
0:  /* Program Name  CS10_3.c */
1:  #include <stdio.h>
2:  #include <stdlib.h>
3:  int main (void)
4:    {
5:       FILE *fptr;
6:       char ch;
7:       clrscr();
8:       if((fptr=fopen("textfile.txt","r"))== NULL)
9:                {
10:                  printf ("\n can't open the file ");
11:                  exit(1);
12:                }
13:      while ((ch=getc(fptr)) !=EOF)
14:        {
15:          printf ("%c",ch);
16:        }
17:     fclose (fptr);
18:     getch();
19:     return 0;
20:  }
```

الشكل رقم ٣-١٠ إختبار مشاكل فتح الملف

وعن هذا البرنامج نوضح ما يلى :

يبدأ البرنامج فى السطر رقم ٨ باختبار الدالة ( ) fopen فاذا فتحت الملف بدون مشاكل تجعل المؤشر يشير الى هذا الملف أما اذا لم تستطيع فتح الملف لأى سبب تعيد الدالة القيمة NULL (\0) وبالتالى فى السطر رقم ٨ نختبر اذا كان الناتج هو

NULL ومعناه لم يتم فتح الملف وتظهر رسالة خطأ ثم ينتهى البرنـامج بالدالـة (1) exit واذا تم فتح الملف بنجاح تنفذ باقى أوامر البرنامج.

## الكتابة والقراءة فى الملف عبارة حرفية كل مرة

فيما يلى نشرح كيفية كتابة البيانات أو قراءة البيانات بإستخدام العبارة الحرفيـة وقبل أن نتابع البرنامج الذى يحقق ذلك نعرض أولا الـدوال المسـتعملة فىكتابـة وقـراءة عبارة حرفية مع الملف

الدالة fputs() وتستخدم لكتابة عبارة حرفية فى ملف

الدالة fgets() وتستخدم لقراءة عبارة حرفية من ملف

الدالة fopen() الدالة العامة لفتح الملف

والبرنـامج الموجـود بالشـكل رقـم ٤-١٠ يسـمح للمسـتخدم بكتابـة كلمـات ويقـوم بتخزينها فى ملف ويظل يستقبل الكلمات ويضعها فى ملف حتى يضغط المستخدم مفتاح الادخال بدون كتابة أى كلمة.

```
0:   /* Program Name  CS10_4.C*/
1:   #include <stdio.h>
2:   #include <conio.h>
3:   main()
4:     {
5:       FILE *fptr;
6:       char string[81];
7:       fptr=fopen("textfile2.txt","w");
8:       while (strlen(gets(string) >0))
9:         {
10:          fputs(string,fptr);
11:          fputs("\n",fptr);
12:        }
```

```
13:     fclose (fptr);
14:  }
```

الشكل رقم ٤-١٠ كتابة عبارة حرفيه فى ملف

وعن هذا البرنامج نوضح ما يلى :

- فى السطر رقم ٧ تم فتح الملف بالطريقة المعتادة لحالة الكتابة
- وفى السطر رقم ٨ الدالة ( ) gets تستقبل من المستخدم عبارة حرفيه والدوارة while تختبرهذه العبارة طالما أن طولها لا يساوى صفر أى لم يضغط المستخدم مفتاح الادخال Enter بدون كتابة عبارة ينفذ سطر رقم ١٠
- فى السطر رقم ١٠ يقوم البرنامج بكتابة العبارة فى الملف الذى يشير اليه المؤشر fptr ثم يعود البرنامج الى الدوارة While فى السطر ٨ وهكذا



الحالة w للكتابة فقط وبالتالى اذا كان الملف موجود من قبل يتم الكتابة فوق بياناته وتفقد البيانات القديمة.

وعند تنفيذ هذاالبرنامج نحصل على النتيجة التالية :

```
write string to the fiel:
the method of write to the file string by string it more quick than
char/char
```

## القراءة من ملف عبارة بعبارة (string by string)

كما يمكن الكتابة فى الملف عبارة حرفية بعبارة حرفية كذلك يمكن القراءة من الملف عبارة حرفية بعبارة حرفية (String by string) وهو أسرع من الكتابة والقراءة حرف بحرف ولكن كل طريقة تستخدم حسب غرض البرنامج وتستخدم الدالة (Fgets) للقراءة من الملف بطريقة عبارة حرفية فى كل مرة والبرنامج الموجود بالشكل رقم ٥-١٠ يقوم بفتح ملف والقراءة منه عبارة بعبارة وطباعة محتويات الملف على الشاشة

```
0:  /* Program Name CS10_5.C*/
1:  #include <stdio.h>
2:  #include <conio.h>
3:  int main(int argc,char *argv[])
4:      {
5:        FILE *fptr;
6:        char string[81];
7:        if(argc !=2)
8:            {
9:              printf ("\n format c>type2 filename");
10:             exit(1);
11:           }
12:       if( (fptr=fopen("argv[1]","r") ) == NULL)
13:         {
14:           printf ("\n can't open this file");
15:           exit(1);
16:         }
17:       while (fgets(string,80,fptr) !=NULL)
18:             printf ("%s",string);
19:       fclose (fptr);
20:       return 0;
21: }
```

الشكل رقم ٥-١٠ قراءة ملف عبارة بعبارة

وعن هذا البرنامج نوضح ما يلى :

- فى السطر رقم ١٧ استخدمنا الدالة fgets() للقراءة من الملف وكما تلاحظ أن لهذه الدالة ثلاث معاملات. الاول هو المتغير string الذى نقرأ فيه البيانات والثانى القيمة ٨٠ ويمثل عدد الحروف المراد قرائتها كل مرة ، والثالث المؤشر الذى يشير الى الملف الذى نقرأ منه
- السطر رقم ١٨ يطبع هذه البيانات ويستمر البرنامج حتى تنتهى محتويات الملف.

## التعامل مع الطابعة والملفات الثابتة

من الموضوعات المهمة اخراج النتائج أو محتويات الملف على الطابعة ويتم ذلك باستخدام دوال الكتابة فى الملف بحيث نكتب متغير ثابت معرف للطابعة وهوالمتغير stdprn ومعناه standard printer مكان اسم الملف المراد الكتابة فيه. والبرنامج الموجود بالشكل رقم ٦-١٠ يوضح كيفية اخراج محتويات ملف على الطابعة

```
0:  /* Program Name CS10_6.C*/
1:  #include <stdio.h>
2:  #include <conio.h>
3:  int main(int argc,char *argv[])
4:    {
5:       FILE *fptr;
6:       char string[81];
7:       if(argc !=2)
8:          {
9:            printf ("\n form is  c:\>type2 filename");
10:           exit(1);
11:         }
12:      if( (fptr=fopen("argv[1]","r") ) == NULL)
13:        {
14:          printf ("\n can't open this file");
15:          exit(1);
16:        }
17:      while (fgets(string,80,fptr) !=NULL)
18:        {
19:           fputs(string,stdprn);
20:           putc('\r',stdprn);
21:        }
22:      flose (fptr);
```

```
23:     return 0;
24:   }
```

الشكل رقم ٦-١٠ أخراج محتويات الملف على الطابعة

وعن هذا البرنامج نوضح ما يلى:

فى السطر رقم ١٩ استخدمنا المتغير الذى يشير الـى الـة الطابعـة وهـو stdpn والدالة fputs() بالصورة fputs(string, stdprn) معناها اكتـب عبـارة حرفيـة (String) على الطابعة المشار اليها بالمتغير stdprn

وهكذا يظل البرنامج يقـرأ محتويـات الملـف ويرسـلها الـى الطابعـة حتـى يصـل البرنامج الى نهاية الملف

## كتابة وقراءة بيانات صحيحة وحقيقية وحرفية

شرحنا فيما سبق كيفية القراءة والكتابة فى الملف باستخدام نوعين من البيانـات وهما متغير من نوع حرف ومتغير من نوع عبارة حرفية وهنا نشـرح كتابـة وقـراءة أنـواع أخرى من أنواع البيانات مثل القيم الصحيحة والقيم الحقيقية ،والدوال المستخدمة لذلك هى :

الدالة fprintf() وتستخدم لكتابة بيانات مختلفة النوع فى ملف
والدالة fscanf() وتستخدم لقراءة بيانات مختلفة النوع من الملف

وهذه الدوال تقوم بنفس عمل الدوال printf(),scanf() ولكن مع الملفات

والبرنامج الموجود بالشكل رقم ٧-١٠ يقوم باستقبال بيانات مختلفة النوع ثم باستخدام الدالة fprintf() يتم كتابة هذه البيانات فى الملف.

```
0:   /* Program Name CS10_7.C*/
1:   #include <stdio.h>
2:   #include <conio.h>
```

```
3:   main ()
4:     {
5:       FILE *fptr;
6:       float height;
7:       int code;
8:       char name[40];
9:      fptr=fopen("txtfile3.txt","a");
10:     do
11:     {
12:        printf ("\n Type name: code number:,and height:");
13:        scanf ("%s",name);
14:        scanf("%d",&code);
15:        scanf("%f",&height);
16:        fprintf(fptr,"%s%d%f",name,code,height);
17:        printf ("\n again(y/n)==>");
18:     }
19:     while (getche()=='y');
20:     fclose(fptr);
21:  }
```

الشكل رقم ٧-١٠ كتابة بيانات مختلفه النوع فى ملف

وعند تنفيذ هذا البرنامج نحصل على النتيجة التالية :

```
Type name: code number:,and height:samy 1 20
again(y/n)==>y
Type name: code number:,and height:hamdy 2 21
again(y/n)==>y
Type name: code number:,and height:nabil  3 23
again(y/n)==>n
```

وعن هذا البرنامج نوضح ما يلى :

- يقوم البرنامج الموجود فى شكل ٧-١٠ فى السطور رقم ١٣و١٤و١٥ باستقبال بيانات هى الاسم والكود والوزن

- في السطر رقم ١٦ يتم كتابة البيانات في الملف الذى يتم فتحه وباستخدام الدالة ()fprintf.
- السطر رقم ١٧ يطبع رسالة (Again (Y/N
- في السطر رقم ١٩ الدوارة While تنتظر استقبال حرف وتختبره فإذا كان الحرف Y تعيد التنفيذ الى السطر رقم ١٠ وتعيد إستقبال بيانات ثم كتابتها في الملف وإذا لم يكن حرف Y ينتهي البرنامج.

## القراءة من ملف باستخدام ( ) fscanf

تقوم الدالة ()fscanf بنفس عمل الدالة ()scanf الا انها تستقبل البيانات من ملف وتكتب بالصورة fscanf (ptr,data) حيث أن المتغير ptr هو الملف المفتوح والمطلوب القراءة منه ،والمتغير data هو المتغير أو المتغييرات التى توضع فيها البيانات المقرؤة.

*مثال*

## الكتابة والقراءة في الملف سجل بسجل (record by record)

تعتبر الكتابة والقراءة في الملف سجل بسجل أنسب طريقة لتطبيقات قواعد البيانات حيث تنشىء سجل (structure) بالموصفات المطلوبة ويتم استعماله سواء للكتابة أو القراءة حسب الرغبة ، والدوال المستعملة لذلك وهى :

الدالة ()fopen التى تستخدم لفتح ملف بصفة عامة

الدالة ()fwrite وتستخدم لكتابة سجل في ملف

الدالة ()fread وتستخدم لقراءة سجل من ملف

وتاخذ الدالة ( )fwrite الصورة التالية :

```
fwrite(&st,sizeof(st),1,ptr);
```

وفيها المتغير &st هو عنوان السجل (structure) المراد كتابته

والماكرو sizeof (st) يعيد حجم السجل المطلوب كتابته بالبايت

والرقم 1 هوعدد السجلات المراد كتابتها كل مرة

والمؤشر ptr هو مؤشر الى الملف المطلوب الكتابة فيه

والبرنامج الموجود بالشكل رقم ٨-١٠ يقوم باستعمال الدالة () fwrite لكتابة سجل بسجل فى ملف كما يظهر ذلك من نتيجة التنفيذ

```
0:  /* Program Name CS10_8.C*/
1:  #include <stdio.h>
2:  int main (void)
3:     {
4:      struct
5:       {
6:          char name[40];
7:          int agnumb;
8:          double height;
9:       } emb;
10:    char numstr[81];
11:    FILE *fptr;
12:
13:    if( (fptr=fopen("agent.rec","wb")) == NULL)
14:      {
15:         printf ("\n can't open this file ");
16:         exit(1);
17:      }
18:    do {
19:       clrscr();
20:       printf ("\n Enter name:");
21:       scanf ("%s",emb.name);
22:       printf ("\n Enter number:");
23:       scanf ("%d",&emb.agnumb);
```

```
24:         printf ("\n Enter height:");
25:         scanf ("%f",&emb.height);
26:         fwrite (&emb,sizeof(emb),1,fptr);
27:         printf ("\n again(y/n)===>");
28:     } while (getch() =='y');
29:   fclose(fptr);
30:   return 0;
31: }
```

الشكل رقم ٨-١٠ الكتابه سجل بسجل

وعند تنفيذ هذا البرنامج نحصل على النتيجة التالية :

```
Enter name:hamdy
 Enter number:321
 Enter height:21
 again(y/n)===>n
```

وعن هذا البرنامج نوضح ما يلى :

فى السطور من رقم ٤ الى السطر رقم ٩ إنشاء سجل و اعلان متغير من نوع هذا السجل هو emb وفى السطر رقم ١٣ يتم فتح ملف للكتابة فيه فى حالة binary mode ومن السطر رقم ١٨ حتى السطر رقم ٢٥ يتم استقبال بيانات السجل من المستخدم وفى السطر رقم ٢٦ يتم كتابة هذا السجل فى الملف باستخدام الدالة fwrite ( ) بالمعاملات التى أشرنا اليها. وهذا البرنامج يصلح ليكون برنامج إضافة بسيط

## القراءة من ملف سجل بسجل

تستخدم الدالة fread() للقراءه من الملف سجل بسجل وتأخذ نفس معاملات الدالة fwrite ( ) كما يلى :

```
fread(&st,sizeof(st),1,ptr )
```

ويتضح ذلك من البرنامج الموجود فى الشكل رقم ٩-١٠ الذى يستخدم الدالة fread() فى قراءة سجلات من ملف ثم عرضها على الشاشة

```
0:  /* Program Name CS10_9.C*/
1:  #include <stdio.h>
2:  void  main (void)
3:      {
4:      FILE *fptr;
6:       struct
7:        {
8:          char name [40];
9:          int numb;
10:         double height;
11:       } emb;
12:     if ( (fptr=fopen("agent.rec","rb") ) ==NULL)
13:       {
14:         printf ("\n can't open this file ");
15:         exit(1);
16:       }
17:     while (fread(&emb,sizeof(emb),1,fptr) ==1)
18:       {
19:        printf ("\n Name: %s",emb.name);
20:        printf ("\n Number: %d",emb.numb);
21:        printf ("\n height: %d",emb.height);
22:       }
23:     fclose (fptr);
24:     return 0;
25:  }
```

شكل رقم ٩-١٠ قراءة ملف سجل بسجل

ونوضح فيما يلى الجديد فى البرنامج الموجود بالشكل رقم ٩-١٠

في السطر رقم ١٧ تقرأ الدالة ( ) fread سجل من الملف بالمعاملات التي أشرنا اليها ولكن الدوارة while تقول طالما أن نتيجة الدالة ()fread تعيد قيمة صحيحة اقرأ من الملف وهي نفس القاعدة السابقة حيث يظل البرنامج يقرأ من الملف طالما توجد فيه سجلات والسطور ١٩ ، ٢٠ ، ٢١ تطبع عناصر السجل على الشاشة.

## الوصول المباشر لسجل معين في ملف random access

في الأمثلة السابقة كانت طرق القراءة سواء حرف بحرف أوعبارة حرفية أو بيانات مختلفة الأنواع أو سجل بسجل تتم بطريقة متتالية أي قراءة سجل ثم الذي يليه وهكذا ، حيث لايمكن الوصول الى سجل الا بعد قراءة السجلات التي قبله أما اذا أردت الوصول مباشرة إلى سجل معين فيجب أن تستخدم الدالة ()fseek التي تقوم بتوجيه مؤشر الملف الىالمكان المطلوب في الملف مباشرة وتأخذ الصورة العامة التالية

int fseek(FILE *stream, long offset, int whence);

وفيها stream مؤشر الى الملف الذي نتعامل معه ، والمتغير whence يحدد مكان بداية الحركة ويأخذ أحد الارقام الثلاثة ٠ أو ١أو ٢ والقيمة ٠ تعني ابدأ البحث من أول الملف والقيمة ١ تعني ابدأ من المكان الحالي لمؤشر الملف والقيمة ٢ تعني أن الحركة تبدأ من نهاية الملف ، والمتغير offset يتم توضيحه في شرح البرنامج التالي والموجود بالشكل رقم ١٠-١٠حيث يسأل المستخدم عن رقم السجل المراد اظهار بياناته ويقوم البرنامج باظهار بيانات هذا السجل مباشرة.

```
0:   /* Program Name CS12_10.C*/
1:   #include <stdio.h>
2:   int main (void)
3:      {
4:        struct
5:            {
6:                char name[40];
7:                int num;
8:                double height;
```

```
9:          }emb;
10:      FILE *fptr;
11:      int recno;
12:      long int offset;
13:      if ( (fptr=fopen("agent.rec","r") ) ==NULL)
14:          {
15:            printf ("can't open this file ");
16:            exit(1);
17:          }
18:      printf("\n ENTER RECORD NO:");
19:      scanf ("%d",&recono);
20:      offset=(recno-1)*sizeof(emb);
21:      if(fseek(fptr,offset,0) !=0)
22:        {
23:           printf ("\n can't move pointer there");
24:           exit(1);
25:        }
26:      fread(&emb,sizeof(emb),1,fptr);
27:      printf ("\n NAME:%s\n",emb.name);
28:      printf ("\n Number:%d\n",emb.num);
29:      printf ("\n Height:%f\n",emb.height);
30:      fclose (fptr);
31:      return 0;
32:  }
```

شكل رقم ١٠-١٠ الوصول الى سجل مباشرة

وعن هذا البرنامج نوضح ما يلى :

السطر رقم ١٨ يسأل المستخدم عن رقم السجل وفى السطر رقم ١٩ تستقبل الدالة ( )scanf رقم صحيح وتخزنه فى المتغير recno و فى السطر رقم ٢٠ يتم حساب عدد البايت المطلوب الذهاب اليها بالمعادلة التالية :

offset=(recno-1)*sizeof(emb)

ومعناها حجم structure الذى يتم معرفته بإستخدام الماكرو sizeof() مضروبا فى رقم السجل مطروح منه القيمة ١ حيث يقف مؤشر الملف في بداية السجل المطلوب ثم باستخدام الدالة ( ) fseek نوجه مؤشر الملف الى السجل المطلوب ونستخدم الدالة ( ) fread فى القراءة فتقرأ السجل المطلوب ثم تتم طباعته على الشاشة

## كتابة وقراءة مجموعات من البيانات (Buffer By Buffer)

كلمة Buffer عبارة عن مساحات متجاورة داخل الذاكرة تستخدم مؤقتاً كمحطات انتقالية لمعالجة المدخلات والمخرجات. كما سبق أن اشرنا فى هذا الفصل أن قراءة الملفات يمكن أن تتم حرف حرف أو عبارة عبارة أو سجلاً سجلاً ، يمكن أيضاً أن تتم على شكل مجموعات وهذه هى الطريقة التي يستخدمها نظام التشغيل فى القراءة والكتابة وتستخدم لهذا الغرض الدوال التالية :

الدالة () read وتقوم بالقراءة من الملف بمقدار عدد من البايت(الحرف)يحدده المبرمج والصورة العامة لهذه الدالة كما يلى :

```
read(int handle, void far *buf, unsigned len,  unsigned *nread);
```

حيث :

| | |
|---|---|
| المتغير handle | عبارة عن رقم يشير الى الملف المراد القراءة منه |
| والمتغير buf | متغير عبارة عن مصفوفة الحروف التى تمثل المخزن المؤقت والذى تخزن به البيانات |
| و المتغير nread | يمثل عدد الحروف المطلوب قراءتها كل مرة |
| | بينما تستخدم الدالة ( ) write لكتابة مجموعة من البيانات فى الملف وتأخذ نفس معاملات الدالة ( ) read. |

وتستخدم الدالة ()open لفتح الملف وتأخذ الصورة العامة التالية :

```
int open(const char *path, [ , unsigned mode ] );
```

حيث :

path يمثل اسم الملف والمسار المراد فتحه

mode هى حالة فتح الملف

وفيما يلى نوضح المعاملات التى تستخدم مع كل حالة من حالات فتح الملف

| المعامل | إستخدامه |
|---|---|
| O_APPEND | تفتح الملف بغرض إضافة بيانات اليه للاضافة وبالتالى تضع مؤشر الملف فى نهاية الملف |
| O_BINARY | تفتح الملف بحالة binary mode |
| O_CREAT | تفتح الملف للكتابة فقط وبالتالى اذا كان الملف موجود من قبل تكتب البيانات الجديدة على البيانات القديمة |
| O_RDONLY | تفتح الملف للقراءة فقط |
| O_RDWR | تفتح الملف للقراءة والكتابة |
| O_TEXT | فى text mode |

يمكن الجمع بين حالتين منطقتيين باستخدام المؤشر | بالصورة التالية

O_RDONLY | O_BINARY كما فى المثال التالى

ومعناها افتح الملف للقراءة فقط وبحالة binary mode

والبرنامج الموجود بالشكل رقم ١١-١٠ يوضح استعمال هذه الدالة فى قراءة بيانات ملف

```
0:  /* Program Name CS10_11.C*/
1:  #include <fcntl.h>
2:  #define BUFFSIZE 512
3:  char  buff[BUFFSIZE];
```

```
4:  int main(int argc,char *argv[])
5:    {
6:    int inhandle,bytes,j;
7:    if (argc != 2)
8:      {
9:        printf ("\n format c:\>serch filename");exit(1);
10:     }
11:    if( (inhandle=open(argv[1],O_RDONLY | O_BINARY)) <0)
12:       {
13:           printf ("\n can't open this file ");
14:           exit(1);
15:       }
16:
17:     while (  (bytes=read(inhandle,buff,BUFFSIZE)) >0)
18:     for (j=0;j<bytes;j++)
19:         putch(buff[j]);
20:     close(inhandle);
21:     return 0;
22: }
```

شكل رقم ١١-١٠ قراءة دفعه من الحروف

وعن هذا البرنامج نوضح ما يلي :

- في السطر رقم ١١ تستخدم لدالة open() لفتح الملف
- وفي السطر رقم ١٧ تقرأ الدالة read من الملف المشار اليه بالرقم inhandle ثم تضع الحروف المقروءة في مصفوفة الحروف buff وتقرأ عدد من البايت مقداره BUFFSIZE. وتعيد العدد الحقيقي في المتغير bytes
- في السطر رقم ١٨ دوارة for تبدأ من صفر الى العدد bytes وهو عدد الحروف الذى أدخل.
- في السطر رقم ١٩ الدالة putch() تطبع عناصر المصفوفة وهي محتويات الملف على الشاشة.

## رسائل الخطأ Error Messages

فى حالة التعامل مع الملفات بطريقـة نظـام التشـغيل وهـى buffer by buffer يمكن لنا تحديد هل تم فتح الملف أم لا وذلك باسـتخدام الدالـة ()open ويتضـح ذلـك من البرنامج الموجود فى الشكل رقم ١٢-١٠

```
0:   /* Program Name CS10_12.C*/
1:   #include <fcntl.h>
2:   #define BUFFSIZE 512
3:    char  buff[BUFFSIZE];
4:    int main(int argc,char *argv[])
5:      {
6:         int inhandle,bytes,j;
7:         if (argc != 2)
8:            {
9:               printf ("\n format c:\>serch filename");exit(1);
10:             }
11:        if( (inhandle=open(argv[1],O_RDONLY | O_BINARY)) <0)
12:            {
13:              perror ("\n can't open this file ");
14:              exit(1);
15:             }
16:        while (  (bytes=read(inhandle,buff,BUFFSIZE)) >0)
17:        for (j=0;j<bytes;j++)
18:              putch(buff[j]);
19:        close(inhandle);
20:        return 0;
21:   }
```

شكل رقم ١٢-١٠ تحديد رسائل الخطأ

وفى البرنامج الموجود فى الشكل رقم ١٢-١٠ تم وضع شرط مع الدالة open () فى السطر رقم ١١ بحيث اذا أعادت الدالة قيمة أقل من صفر فهذا يدل على أن الدالة لم تسطيع فتح الملف وبالتالى يقوم البرنامج بطباعة رسالة خطأ وذلك فى السطر رقم ١٣ وينتهى البرنامج

## كتابة مجموعات من البيانات buferr by buffer

المقصود بذلك هو استقبال كمية كبيرة من البيانات ثم تسجيلها مرة واحدة فى الملف وهذا الاسلوب مفيد مع برامج كتابة الرسائل حيث يسمح للمستخدم بالكتابة وفى النهاية يخزن البيانات فى الملف والبرنامج الموجود بالشكل رقم ١٣-١٠ يوضح هذه الطريقة

```
0:  /* Program Name CS10_13.C*/
1:  #include <fcntl.h>
2:  #include <stat.h>
3:  #define BUFFSIZE 4096
4:  char  buff[BUFFSIZE];
5:  int main(int argc,char *argv[])
6:     {
7:          int inhandle,outhandle,bytes,j;
8:          if (argc != 3)
9:              {
10:                 printf ("\n format c:\>copy2 file1 file2");exit(1);
11:             }
12:         if( (inhandle=open(argv[1],O_RDWR | O_BINARY)) <0)
13:             {
14:                 printf ("\n can't open INPUT file ");
15:                 exit(1);
16:             }
17:l  if((outhandle=open(argv[2],O_CREAT|O_BINARY|S_WRITE))<0)
18:             {
19:                 printf ("\n can't open OUTPUT file ");
```

```
20:            exit(1);
21:         }
22:      while (  (bytes=read(inhandle,buff,BUFFSIZE)) >0)
23:         {
24:            write (outhandle,buff,bytes);
25:         }
26:      close(inhandle);
27:      close(outhandle);
28:      return 0;
29:  }
```

الشكل رقم ١٣-١٠ كتابة الحروف دفعه دفعه

وعن هذا البرنامج نوضح ما يلى :

- يقوم البرنامج بفتح ملفين ، فى السطر رقم ١٠ يتم فتح الملف الأول للقراءة binary mode وفى السطر رقم ١٧ يتم فتح الملف الثانى للكتابة بحالة binary mode أيضاً
- فى السطر رقم ٢٢ يتم القراءة من الملف الأول بالدالة ()read وفىالسطر رقم ٢٤ يتم الكتابة فى الملف الثانى ويستمر البرنامج فى القراءة من الملف الأول والكتابة فى الملف الثانى حتى ينتهى الملف الاول ،وبالتالى يتم انشاء ملف جديد يحتوى على نفس محتويات الملف الاول وهى نفس فكرة أمر Copy الذى يستخدمه نظام التشغيل DOS

## إرسال المخرجات الى الطابعة

ارسال المخرجات إلى الطابعة باستخدام الدالة ( ) write ، فبدلا من الكتابة فى ملف نكتب الرقم الخاص بالطابعة وهو الرقم ٤ وهذا الرقم محجوز للطابعة الموصلة بالجهاز ويتضح ذلك من البرنامج الموجود بالشكل رقم ١٤-١٠

```
0:  /* Program Name CS10_14.C*/
1:  #include <fcntl.h>
2:  #define BUFFSIZE 512
```

```
3:   char  buff[BUFFSIZE];
4:   int main(int argc,char *argv[])
5:     {
6:         int inhandle,bytes,j;
7:         if  (argc != 2)
8:           {
9:             printf ("\n format c:\>serch filename");exit(1);
10:          }
11:        if( (inhandle=open(argv[1],O_RDONLY | O_BINARY)) <0)
12:          {
13:            printf ("\n can't open this file ");
14:            exit(1);
15:          }
16:        while (  (bytes=read(inhandle,buff,BUFFSIZE)) >0)
17:              write (4,buff,bytes);
18:        close(inhandle);
19:        return 0;
20:   }
```

شكل رقم ١٤-١٠

والجديد فى البرنامج أن الدالة write والموجودة بالسطر رقم ١٧ تكتب فى الملف رقم ٤ وهذا الرقم محجوز لآلة الطباعة وبالتالى تخرج النتائج على الطابعة

## متى تستخدم كل طريقة من الطرق السابقة

وفى نهاية هذا الفصل نسأل متى تستخدم كل طريقة من الطرق السابقة للتعامل مع الملفات وللرد على هذا التساؤل نقول :

*اذا أردت أن تكتب برنامج معالج كلمات word processor يفضل استعمال طريقة Beffer By Buffer. وهىالكتابة والقراءة بمقدار المخزن المؤقت

*اذا أردت عمل dbase system تستعمل طريقة القراءة والكتابة باستخدام السـجلات structure By Strudure.

*أذا أردت عمل برنامج يتعامل مع الحروف مثل عد حـروف ملـف مثـلا أو قياس سـرعة كاتب نستعمل طريقة القراءة والكتابة حرف بحرف

## حالتى الكتابة والقراءة من ملف text mode,binary mode

يمكن تسجيل البيانات فى الملفات في صورتين

*الصورة الاولى* تسـمى text mode وهـى تسـجيل جميـع البيانـات فـى شـكل حروف حتى الارقام كل رقم يأخذ بايت وهى طريقة النظام dos ، ويتـم تمثيـل علامـة السـطر الجديـد (CR/LF) carriage return-linefeed وتمثيل نهاية الملف (EOF) بالكود 1A

*الصورة الثانية* تسمى binary mode وهى تسجيل جميع البيانات فـى شـكلها الطبيعى الارقام تسجل كأرقام والحروف كحروف وهـى طريقـة نظام التشغيل unix وفيها يتم تمثيل علامة السطر الجديد حـرف واحد linefeed LF.

من هذه المقارنـة تجـد أن الملـف اذا كتـب فـى حالـة text و قـرىء فـى حالـة binary فأن عدد الحروف سيختلف، ويفضل اسـتعمال حالـة binary اذا كـان البرنـامج يتعامل مع الارقام.

ويتم تحديد حالة فتح ملف بطريقة text mode أو binary mode باسـتخدام الداله ()fopen ففى حالة binary mode نضـع حـرف b بجـانب الحـرف الـدال علـى الفتح كما يلى :

```
fopen("filename","rb");
```

أما فى حالة text mode فتكتب الدالة كما سـبق فـى البـرامج بـدون حـرف b بالصورة التالية :

```
fopen("filename","r");
```

وهى الحالة الافتراضية

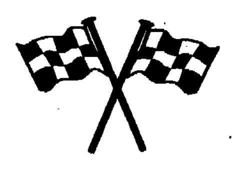
والبرنامج الموجود فى الشكل رقم ١٥-١٠ يوضح الفرق بين الحالتين وذلك بفتح ملف فى binary mode من الملفات الموجودة على القرص والمكتوبة فى text mode وطبع عدد حروفه ومن نظام التشغيل DOS وبالامر dir نقارن عدد الحروف من البرنامج ونظام التشغيل لنرى الفرق.

```
0:   /* Program Name CS10_15.C*/
1:   #include <stdio.h>
2:   #include <conio.h>
3:   main(int argc ,char *argv[])
4:     {
5:       FILE *fptr;
6:       char string[81];
7:       int count=0;
8:       if (argc !=2)
9:          {
10:            printf("\n format  c>charcnt filename");
11:            exit(1);
12:          }
13:      if((fptr=fopen(argv[1],"rb"))==NULL);
14:         {
15:            printf ("can't open this file");
16:            exit(1);
17:         }
18:      while (getc(fptr) !=EOF)
19:            count++;
20:     fclose (fptr);
21:     printf ("\n file %s contains %d chrcters ",argv[1],count);
22:   }
```

الشكل رقم ١٥-١٠

وعن هذا البرنامج نوضح ما يلي :

البرنامج الموجود بالشكل رقم ١٥-١٠ هونفس برنامج عـد الحـروف السـابق والموجود بالشكل رقم ٤-١٠ والفرق بينهما أن هذا البرنـامج يقـوم بفتـح الملـف فى binary mode وذلك في السطر رقم ١٣ وبالتالى يختلف عدد الحروف.



# الفصل الحادى عشر
# دوال مهمة للمبرمج

*فى هذا الفصل تتناول مجموعه من الدول التى تساعد فى تحسين أداء البرنامج حيث تتناول الدول التاليه:-*

- *دالة تحديد احداثيات نافذة على الشاشة*
- *نسخ جزء من الشاشة إلى موضع آخر*
- *حفظ جزء من الشاشة فى متغير*
- *استرجاع الجزء المحفوظ من متغير*
- *تغير درجة اضاءة الحرف*
- *تخزين مكان المؤشر فى متغير* **Wherex(),wherey()**
- *التعامل مع الالوان والصوت*
- *برنامج محرر نصوص*

## دالة تحديد احدثيات نافذة على الشاشة

من الوظائف المهمة تحديد جزء معين على الشاشة بحيث يمكن اجراء عمليات معينة على هذا الجزء فقط مثل تغيير ألوان الكتابة أو الخلفية لهذا الجزء. ويتم ذلك بتحديد احداثيات هذا الجزء باستخدام الدالة window تأخذ الدالة window الشكل التالى:

window(L,T,R,B)

حيث أن المتغيرات L تعنى LEFT وهو الطرف الايسر للنافذه،T تعنى TOP وهو الطرف العلوى للنافذه ، R : RIGHT أى الطرف الايمن ، B : BOTTOM أى أسفل النافذه أى الاحدثيات الأربعة للجزء المحدد وهى احداثى نقطة الركن الشمالى العلوى واليمين السفلى.

والبرنامج الموجود بالشكل رقم ١-١١ يقوم بتحديد احداثيات نافذة ثم استعمال الدوارة for لتكرار طباعة كلمة Allah داخل هذه النافذة فتعطى النتيجة الموجود ة فى الشكل رقم ٢-١١

```
0:  /*Program Name CS11_1.C*/
1:  #incclude <conio.h>
2:  #define LEFT 10
3:  #define TOP 8
4:  #define RIGHT 52
5:  #define BOT 21
7:  void main (void)
8:      {
9:          int j;
10:         window(LEFT,TOP,RIGHT,BOT);
11:         textcolor(RED);
12:         textbackground(GREEN);
13:         for (j=0;j<200;j++)
14:             {
```

```
15:                 cputs(" ALLAH ");
16:                 delay(10);
17:             }
18:         gotoxy(15,8);
19:         cputs("  THE GOD  ");
20:
21:         getch();
22:   }
```

شكل رقم ١-١١ استخدام الدالة window لتحديد احداثيات نافذة

وعن تنفيذ هذا البرنامج نحصل على النتيجة التالية :

```
H  ALLAH  ALLAH  ALLAH  ALLAH  ALLAH  ALLAH
 ALLAH  ALLAH  ALLAH  ALLAH  ALLAH  ALLAH
 ALLAH  ALLAH  ALLAH  ALLAH  ALLAH  ALLAH
ALLAH  ALLAH  ALLAH  ALLAH  ALLAH  ALLAH  A
LLAH  ALLAH  ALLAH  ALLAH  ALLAH  ALLAH  AL
LAH  ALLAH  ALLAH  ALLAH  ALLAH  ALLAH  ALL
AH  ALLAH  ALLAH  ALLAH  ALLAH  ALLAH  ALLA
H  ALLAH  ALLA   THE GOD  LAH  ALLAH  ALLAH
 ALLAH  ALLAH  ALLAH  ALLAH  ALLAH  ALLAH
 ALLAH  ALLAH  ALLAH  ALLAH  ALLAH  ALLAH
ALLAH  ALLAH  ALLAH  ALLAH  ALLAH  ALLAH  A
LLAH  ALLAH  ALLAH  ALLAH  ALLAH  ALLAH  AL
LAH  ALLAH  ALLAH  ALLAH  ALLAH  ALLAH  ALL
AH  ALLAH  ALLAH  ALLAH
```

شكل رقم ٢-١١

وعن هذا البرنامج نوضح ما يلى :

- يشتمل السطر رقم ١٠ على الدالة () Window وبها المعاملات LEFT, TOP, RIGHT, BOT التى تم تعريفها فى أول البرنامج ومعناها تحديد نافذة بهذة الاحداثيات.
- وفى السطرين رقم ١١ و ١٢ تقوم الدالتين ()textcolo و ()textbackground بتغيير لون الكتابة ولون الخلفية.
- فى السطر رقم ١٣ الدوارة for للتكرار ٢٠٠ مرة.

- في السطر رقم ١٥ الدالة ()cputs التي تطبع كلمه ALLAH وهي نفس عمل الدالة ()puts ولكنها تكتب داخل النافذة التي تم تحديدها بالالوان والقيم المحددة مسبقاً.

## نسخ جزء من الشاشة إلى موضع آخر

يمكن نسخ أى مساحة من الشاشة إلى مكان أخر على الشاشة وذلك باستخدام الدالة ()movetext. وتظهر بالصورة الآتية :

**movetext (LEFT, TOP, RIGHT, BOT, DESLEFT, DESTOP) ;**

حيث المتغيرات LEFT , TOP, RIGHT, BOT هي احداثيات المساحة المراد نسخها

والمتغيران DESLEFT, DESTOP هما احداثيات النقطة المراد النسخ اليها ، والبرنامج الموجود بالشكل رقم ٣-١١ يقوم بالكتابة على الشاشة ثم يقوم باستعمال الدالة ()movetext لنسخ مساحة إلى مكان اخر من الشاشة كما يتضح من نتيجة التنفيذ الموجودة فى الشكل رقم ٤-١١

```
0:  /*Program Name CS11_3.C*/
1:  #include <conio.h>
2:  #define LEFT 26
3:  #define TOP 7
4:  #define RIGHT 65
5:  #define BOT 20
6:  #define DESLEFT 1
7:  #define DESTOP 1
8:  #define NUMCOLORS 16
9:  #define HEIGHT (BOT-TOP+1)
10: void fancy_box(int x1,int y1,int x2,int y2)
11:     {
12:         int i;
13:         gotoxy(x1,y1);putch(201);
14:         for(i=x1+1;i<x2;i++) putch(205);
```

```
15:         putch(187);
16:     for(i=y1+1;i<y2;i++) {
17:        {  gotoxy(x1,i);putch(186);
18:           gotoxy(x2,i);putch(186);
19:        }
20:     gotoxy(x1,y2);putch(200);
21:     for (i=x1+1;i<x2;i++) putch(205);
22:         putch(188);
23:  }

24: void main (void)
25:   {
26:     int j;
28:     clrscr();
29:     fancy_box(LEFT,TOP,RIGHT,BOT);
30:     window(LEFT+1,TOP+1,RIGHT-1,BOT-1);
31:     /*textcolor(RED);*/
32:     textbackground(GREEN);
33:     for (j=0;j<98;j++)
34:        {
35:          textcolor(j % NUMCOLORS);
36:          cputs(" ALLAH ");
37:           /*delay(10); */
38:        }
39:     /*delay(200); */
40:     movetext(LEFT,TOP,RIGHT,BOT,DESLEFT,DESTOP);
41:     getch();
42:     movetext(LEFT,TOP,RIGHT,BOT,DESLEFT+27,DESTOP);
43:     getch();
44:  }
```

شكل رقم ٣-١١

وعن هذا البرنامج نوضح ما يلى :

- من السطر رقم ١٠ حتى السطر رقم ٢٣ يتم انشاء دالة لرسم مستطيل مزدوج الخط
- وفى السطر رقم ٢٩ يتم استدعاء هذه الدالة ، وفى السطر رقم ٣٠ تحدد الدالة ()window إحداثيات نافذة وباستخدام دالة ()fancy_box نرسم مستطيل حول هذه النافذه
- وفى السطر رقم ٤٠ تقوم الدالة ()movetext بنسخ المساحة المحددة بالاحداثيات LEFT,TOP,RIGHT,BOT والمعرفة فى أول البرنامج الى النقطة التى احداثياتها DESLEFT,DESTOP والمعرفة أيضا فى أول البرنامج ثم تكرر نفس العملية فى احداثيات أخرى فى السطر رقم ٤٢

شكل رقم ٤-١١

## حفظ جزء من الشاشة فى متغير

يمكن حفظ أى مساحة من الشاشة بمحتوياتها فى متغير لاسترجاعها عند الحاجه اليها ويتم ذلك باستخدام الدالة ()gettext والتى تأخذ الصورة العامة التالية :

```
gettext(LEFT,TOP,RIGHT,BOT,buff);
```

حيث المتغيرات LEFT,TOP,RIGHT,BOT هي احداثيات المساحة المراد حفظها. والمتغير buff متغير من نوع مصفوفة نستعمله فى تخزين المساحه المطلوب حفظها

## استرجاع الجزء المحفوظ

بعد حفظ أى جزء من الشاشة فى متغير باستخدام الدالة gettext() يمكن استرجاع هذا الجزء بالدالة puttext() والتى تأخذ الصورة التالية :

puttext(LEFT,TOP,RIGHT,BOT,buff);

كما تلاحظ أن الدالة puttext تاخذ نفس المعاملات التى تأخذها الدالة gettext()

والبرنامج الموجود بالشكل رقم ٥-١١ يقوم باستخدام كلا من الدالة gettext() والدالة puttext() فى حفظ جزء من الشاشة ثم مسح الشاشة ثم استرجاع هذا الجزء

```
0:  /*Program Name CS11_5.C*/
1:  #include <conio.h>
2:  #define LEFT 1
3:  #define TOP 1
4:  #define RIGHT 80
5:  #define BOT 25
6:  #define DESLEFT 1
7:  #define DESTOP 1
8:  #define NUMCOLORS 16
9:  #define HEIGHT (BOT-TOP+1)
10: int buff[80][25];
11: void fancy_box(int x1,int y1,int x2,int y2)
12:    {
13:      int i;
14:      gotoxy(x1,y1);putch(201);
15:      for(i=x1+1;i<x2;i++) putch(205);
16:      putch(187);
17:      for(i=y1+1;i<y2;i++)
```

```
18:          { gotoxy(x1,i);putch(186);
19:             gotoxy(x2,i);putch(186);
20:          }
21:     gotoxy(x1,y2);putch(200);
22:     for (i=x1+1;i<x2;i++) putch(205);
23:       putch(188);
24:   }
25: void main (void)
26:   {
27:     int x,y,j;
28:     fancy_box(LEFT,TOP,RIGHT,BOT);
29:     window(LEFT+1,TOP+1,RIGHT-1,BOT-1);
30:     textcolor(RED);
31:     textbackground(GREEN);
32:     for (j=0;j<200;j++)
33:        {
34:              textcolor(j % NUMCOLORS);
35:              cputs(" ALLAH ");
36:          }
37:     gettext(LEFT,TOP,RIGHT,BOT,buff);
38:     x=wherex();
39:     y=wherey();
40:     clrscr();
46:     getch();
47:     clrscr();
48:     puts("restore screen ....");
49:     getch();
50:     puttext(LEFT,TOP,RIGHT,BOT,buff);
51:     getch();
52:     clrscr();
53:     puts("restore screen ....");
54:     getch();
55:     puttext(LEFT,TOP,RIGHT,BOT,buff);
56:     gotoxy(x,y-1);
```

```
57:     getch();
58: }
```

شكل رقم ٥-١١ استخدام دوال حفظ واسترجاع الشاشة

وعن هذا البرنامج نوضح ما يلى :

- فى السطر رقم ٣٧ تقوم الدالة gettext() بحفظ المساحة المحددة بالاحداثيات LEFT,TOP,RIGHT,BOT فى المتغير buff وجميع هذه المتغيرات تم تعريفها فى أول البرنامج
- وفى السطر رقم ٤٠ يتم مسح الشاشة باستخدام الدالة clrscr وفى السطر رقم ٤٦ تجعل الدالة getch() البرنامج ينتظر الضغط على أى مفتاح
- فى السطر رقم ٥٠ يتم استعادة المساحة المخزنة فى المتغير buff مرة أخرى باستخدام الدالة puttext() وهكذا باقى البرنامج

## تغيير درجة اضاءة الحرف

المقصود بتغيير درجة اضاءة الحرف جعلها مضيئة أو خافتة أو عادية ويستخدم لهذا الغرض الدوال التالية :

| الدالة | وظيفتها |
|---|---|
| highvideo() | تجعل الحرف مضىء |
| lowvideo() | تجعل الحرف خافت |
| normvideo() | تعيد اضاءة الحرف الى الحالة المعتادة |

كما توجد مجموعة من الدوال المفيدة التى تؤدى خدمات مختلفة معرفة كلها فى الملف conio.h

نذكر منها :

| الدالة | وظيفتها |
|---|---|
| delline() | تحذف سطر من سطور النص |
| insline() | تضيف سطر بين سطور النص |
| textattr() | تغيير طبيعة الحروف من حيث الالوان وطبيعة الظهور |
| textmode() | تغيير حالة كتابة النصوص الى حالة أخرى |

والبرنامج الموجود فى الشكل رقم ٦-١١ مثال مصغر لبرنامج محرر سطور (Editor) حيث يسمح لك بكتابة سطور داخل نافذه محددة الاحداثيات كما يسمح ببعض العمليات المشهورة مثل مسح سطر أو اضافة سطر أو الرجوع الى العملية السابقة وذلك باستخدام الدوال التى ذكرناها

```
0:  /* Program Name CS11_6.C*/
1:  #include <conio.h>
2:  #define LEFT 10
3:  #define TOP 8
4:  #define RIGHT 50
5:  #define BOT 21
6:  #define WIDTH (RIGHT-LEFT+1)
7:  #define TRUE 1
8:  #define ESC 27
9:  #define HEIGHT (BOT-TOP+1)
10: #define L_ARRO 75
11: #define R_ARRO 77
12: #define U_ARRO 72
13: #define D_ARRO 80
14: #define INS 82
15: #define DEL 83
16: #define ALT_H 35
17: #define ALT_C 46
18: #define ALT_U 22
19: #define ALT_L 38
20:  int buff[WIDTH][HEIGHT];
```

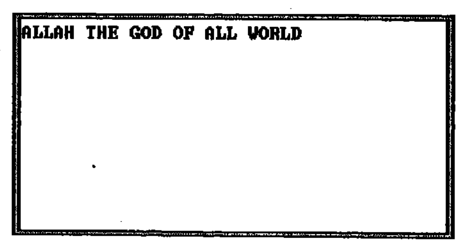
```
21: void fancy_box(int x1,int y1,int x2,int y2)
22: {
23:    int i;
24:    gotoxy(x1,y1);putch(201);
25:    for(i=x1+1;i<x2;i++) putch(205);
26:    putch(187);
27:    for(i=y1+1;i<y2;i++) {
28:    gotoxy(x1,i);putch(186);
29:    gotoxy(x2,i);putch(186);
30:    }
31:    gotoxy(x1,y2);putch(200);
32:    for (i=x1+1;i<x2;i++) putch(205);
33:    putch(188);
34:    }
35: void main (void)
36:       {
37:       int x,y,j;
38:       char key;
39:       x=1;y=1;
40:       clrscr();
41:             fancy_box(LEFT,TOP,RIGHT,BOT);
42:        printf("\n\n   INS: inseret line, DEL: delete line ESC:EXIT");
43:        printf ("\n   ALT_u undo,ALT_H high intensity,ALT_C color
                                                   ENG.AZAB ");
44:       window(LEFT+1,TOP+1,RIGHT-1,BOT-1);
45:       textbackground(GREEN);
46:    while ( (key=getch() ) !=ESC)
47:                    {
48:                    if (key==0)
49:                     {
50:                     switch(getch())
51:                      {
52:                      case L_ARRO:
53:                        if (x>1)
```

```
54:                     gotoxy(--x,y);
55:                     break;
56:              case R_ARRO:
57:               if (x<WIDTH)
58:                     gotoxy(++x,y);
59:                     break;
60:              case U_ARRO:
61:                if (y>1)
62:                     gotoxy(x,y--);
63:                     break;
64:              case D_ARRO:
65:                    if(y<HEIGHT)
66:                     gotoxy(x,++y);
67:                     break;
68:              case INS:
69:              gettext(LEFT+1,TOP+1,RIGHT-1,BOT-1,buff);
70:                    insline();
71:                    break;
72:               case DEL:
73:              gettext(LEFT+1,TOP+1,RIGHT-1,BOT-1,buff);
74:                    delline();
75:                    break;
76:                case ALT_H:
77:                    highvideo();
78:                    break;
79:              case ALT_L:
80:                    lowvideo();
81:                    break;
82:              case ALT_C:
83:                    textcolor(getch()-'0');
84:                    break;
85:              case ALT_U:
86:              puttext(LEFT+1,TOP+1,RIGHT-1,BOT-1,buff);
87:                    break;
```

```
88:                 case '\r':
89:                     printf ("\n");
90:                     break;
91:             } /* end of switch   */
92:
93:         } /* end if  */
94:     else
95:       {
96:       putch(key);
97:       x=wherex();
98:       y=wherey();
99:       }
100:    }  /*end of while   */
101:  } /* end of main () */
```

شكل رقم ٦-١١ محرر سطور بسيط

و عند تنفيذ : البرنامج نحصل على النتيجة التالية




INS: inseret line, DEL: delete line ESC:EXIT
ALT_u undo,ALT_H high intensity,ALT_C color ENG.AZAB

شكل ٧-١١

وعن هذا البرنامج نوضح ما يلى :

- فى السطر رقم ٤١ الدالة fancybox() ترسم مستطيل يحدد المساحة التى يكتب فيها المستخدم.

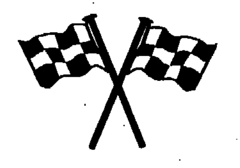
- السطر رقم ٤٢ و ٤٣ لطباعة التعليمات الخاصة بمحرر السطور
- وفي السطر رقم ٤٤ الدالة window() تقوم بتحديد مساحة الكتابة
- ثم في السطر رقم ٤٦ تبدأ الدالة getche() في استقبال حرف

في السطر رقم ٤٨ جملة if تختبر هل الحرف المدخل ممتد (مفتاح وظائف) (راجع الفصل السابع) أم مفتاح عادى اذا كان مفتاح من مفاتيح الوظائف تستقبل الدالة الرقم الثاني المميز له وفي السطر رقم ٥٠ جملة switch تختبر الكود الثاني المميز للحرف الممتد وفيها الحالة الأولى والثانية لتوظيف مفتاحى الاسهم الايمن والايسر والحالة الثالثة و الرابعة لتوظيف مفتاحى الاسهم العلوى والسفلى

- السطر رقم ٦٨ توظيف المفتاح ins باستدعاء الدالة insline()
- السطر رقم ٧٢ توظيف المفتاح del باستدعاء الدالة delline()
- السطر رقم ٧٦ توظيف مفتاحى alt_h لجعل الكتابة مضيئة باستخدم الدالة highvideo() ، وبالمثل السطر ٧٩ يجعل الكتابة خافتة باستخدام الدالة lowvideo()
- في السطر رقم ٨٢ الدالة تستقبل getch() رقم من المستخدم. هذا الرقم يدل على اللون المطلوب ثم تضع رقم اللون كمعامل للدالة textcolor() بعد طرح كود ، حتى يطابق الرقم المطلوب.
- يقوم السطر رقم ٨٥ بالوظيفة undo ومعناها الرجوع عن أخر عملية تمت ، ويتم ذلك عن طريق حفظ الشاشة بعد كل عملية بالدالة gettext()

تنفذ جميع حالات التحرير (تحريك السهم أو الحذف أو النسخ) اذا كان المفتاح الذى ضغط عليه المستخدم أحد مفاتيح الوظائف أما اذا كان المفتاح

مفتاحاً عادياً فيظهر الحرف على الشاشة ويعاد تنفيذ الدوارة ما لم يضغط المستخدم مفتاح ESC.





# الباب الثانى عشر
# الرسم فى لغة C

الرسم من الموضوعات الجذابة في معظم اللغات ولمعظم هواة البرمجة، ولغة C لها نصيب كبير من دوال الرسم بل ان شئت قل أكبر نصيب بين اللغات ، فهى تحتوى على جميع دوال أدوات الرسم، وفى هذا الفصل نناقش هذه الدوال ونتعرف على الموضوعات التالية :

- تهيئة الشاشة لحالة الرسم
- تحديد حالة الرسم المناسبة تلقائى
- تحديد الاخطاء الناتجة فى حالة الرسم
- رسم الخطوط بأنوعها مع تغير الالوان
- رسم قطع الناقص ومتعدد الخطوط
- التلوين و(التظليل ) والاشكال المختلفة للتظليل
- الرسم البيانى ذو البعدين والأبعاد الثلاثة( 3-DIM, 2-DIM)
- تحريك الرسوم
- استخدام خطوط الكتابة

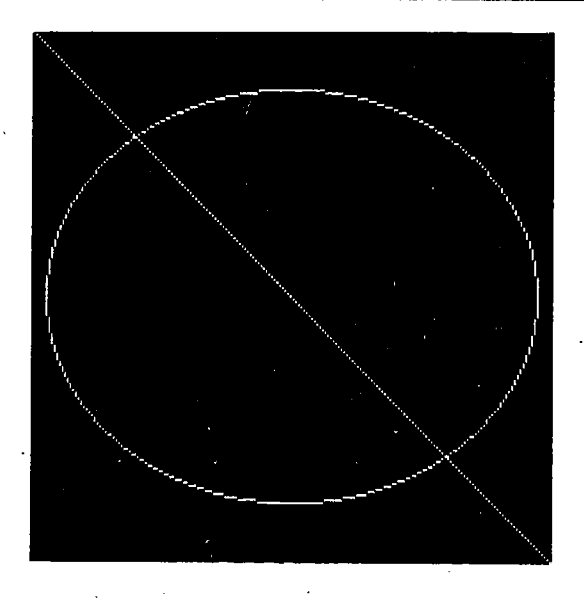
تستخدم لغة C مجموعة كبيرة من دوال الرسم ولكن قبل أن نبدأ في شرح واستعمال هذه الدوال نوضح كيفية تهيئة الجهاز لحالة الرسم حتى يقبل تنفيذ دوال الرسم

## تهيئة الشاشة لحالة الرسم Initialization Graphics Mode

شرحنا في الفصول السابقة كثير من الدوال مثل دالة طباعة البيانات على الشاشة وكذلك دوال تغير لون الكتابة ولون الخلفية ، وجميع هذه الدوال تعمل في حالة النصوص (text mode) ولكن لاستعمال دوال الرسم لابد من تحويل الشاشه من حالة النصوص الى حالة الرسم ( grapdics mode) ولتوضيح ذلك نتابع البرنامج الموجود بالشكل رقم ١-١٢ الذى يقوم بتحويل الشاشة الى حالة الرسم ثم يرسم خط ودائرة ويتضح ذلك من الشكل ٢-١٢

```
0:  /*Program Name CS12_1.C*/
1:  #include <graphics.h>
2:  void main (void)
3:     {
4:        int driver,mode;
5:        int x1=0,y1=0;
6:        int x2=199,y2=199;
7:        int xc=100,yc=100;
8:        int radius=40;
9:        driver=CGA;
10:       mode=CGAC0;
11:       initgraph(&driver,&mode," ");
12:       line(x1,y1,x2,y2);
13:       circle(xc,yc,radius);
14:       getch();
15:       closegraph();
16:    }
```

شكل ١-١٢برنامج تحويل الشاشه إلى حالة الرسم ورسم خط ودائرة



شكل ٢-١٢ رسم خط ودائرة

وقبل شرح هذا البرنامج نوضح ما يلى :

من المعروف أن للشاشات كارت يركب داخل الجهاز وتوصل به الشاشة وهو ما يسمى بكارت الشاشة ( Display Card) وتوجد أنواع كثيرة من هذه الكروت من أشهرها :

كارت شاشة من نوع CGA (Color Graphics Adptor)

كارت شاشة من نوع EGA (Enhanced Graphics Adptor)

كارت شاشة من نوع VGA (Vidio Array Adptor)

و فى حالة الرسم (فى لغة C) يحتاج كل نوع من هذه الكروت لملف يسمى driver ليهيء الشاشة للرسم وهذا الملف يأتى مع مترجم لغة C داخل الفهرس المسمى bgi بالاسماء HERC.BGI ,EGAVGA.BGI, CGA.BGI،وكل ملف من هذه الملفات خاص بنوع شاشة (وكارت الشاشة ) فمثلاً الملف EGAVGA خاص

بالشاشات EGA والشاشـات VGA . ويجـب أن نكتـب اسـم الملـف الموافـق لكـارت الجهاز الذى نستعمله.

والأن نعود لشرح البرنامج

- من السطر رقيم ٤ إلى السطر رقم ٨ اعلان عن مجموعه متغيّرات.
- في السطر رقم ٩ نضع فى المتغير الصحيح driver رقم ،هذا الرقم معرف في المتغير CGA وهو يشير الى اسم الملف المطلوب لتهيئـة الشاشـة والمتوافـق مع الشاشة المستعملة ، والمتغير CGAC0 يحدد حالة رسـم وجميـع الحـالات معرفة في الملف graphics.h بمجموعة متغيرات.
- و في السطر رقـم ١١ الدالـة ()initgraph تقـوم بتهيئـة الشاشـة لحالـة الرسـم ومعاملاتها كما يلي:

  الاول عنوان المتغير driver الذى يشير الى نوع الكارت المركب.

  الثاني عنوان المتغير mode الذى يحدد الحالة المطلوبه.

  المعامل الثالث للدالة هو المسار الموجود فيه ملف التهيئـة ( CGA.BGI أو EGAVGA.BGI) حسب نوع الشاشة.

- والسطور من ٤ الى ١١ تعتبر جزء ثابت للتحويل الى حالة الرسم.
- وفي السطر رقم ١٢ نستخدم دالـة ()line لرسـم خـط لـه نقطتـان ، إحداثيـات النقطة الأولى هما x1,y1 وإحداثي النقطة الثانية هما x2,y2

وفي السطر رقم ١٣ الدالة ()circle لرسم دائـرة حيـث xc,yc مركـز الدائـرة والمتغير radius نصف قطـر الدائـرة وجميـع هـذه المتغـيرات معرفـة فـى أول البرنامج.

- وفى السطر رقم ١٥ الدالة ()closegraph تقوم بإغلاق حالة الرسم والرجـوع الى حالة كتابة النصوص العادية ولو لم نستخدم هذه الدالـة سـتجد أن أي كـلام

يكتب على الشاشة يكتب و كأنه رسم ويصبح المؤشر غير مرئـي وذلـك لأنـك مازلت في حالة الرسم.

## تحديد حالة الرسم تلقائياً Auto Initialization Detect

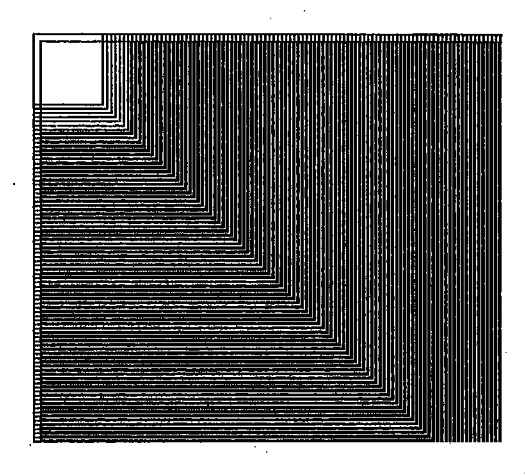
في البرنامج السابق رأينا أنه لاستعمال دوال الرسم لابد من تحديد نـوع كـارت الشاشة وبالتالي نوع الملف المستخدم لتهيئة الشاشة للرسم وكذلك تحديد الحالة (mode) التي نريدها من حـالات الرسـم وهـذا الأمر مجهد ولـه عيـوب مـن أهمهـا أن البرنامج المصمم ليعمل مـع كـارت شاشة معين لايعمـل مع كـارت شاشـة أخـر فمثلا البرامج المكتوبة لتعمل مع شاشة من نوع CGA لاتعمل مع شاشـة مـن نـوع EGA أو VGA والعكس صحيح. وهذا أمر متعب حيث كانت البرامج القديمـة تسأل المستخدم أولا عن نوع الشاشة ثم تقـوم بتهيئـة الجهاز ليعمـل مـع الكـارت المناسـب. ولكـن مـن الأفضل ترك مهمة تحديد نوع كارت الشاشة وتحديد حالة التشـغيل للبرنـامج. وهـذا مـا نراه في الشكل رقم ٣-١٢ الذي يتولى تحديد كارت الشاشة والحالـة المناسـبة لهـا ثـم يقوم باستعمال دالة رسم مستطيل ودالة الصوت لرسم مجموعـة مستطيلات كمـا تظهـر النتيجة كما في الشكل رقم ٤-١٢

```
0:  /*Program Name CS12-3C*/
1:  #include<graphics.h>
2:  void main(void)
3:    {
5:       int i;
6:       int driver =DETECT;
7:       int mode;
8:       int maxx,maxy;
9:       int left,top,right,bot;
10:      initgraph(&driver,&mode,"");
11:      maxx=getmaxx();
12:      maxy=getmaxy();
13:      left=top=0;
```

```
14:    right=maxx;
15:    bot=maxy;
16:    for(i=0;i<400;i+=3)
17:       {
18:          rectangle(left,top,50+i,50+i);
19:          rectangle(left+5,top+5,70+i,70+i);
20:          sound(i*40);
21:          delay(10);
22:          nosound();
24:       }
25:    getch();
26:    closegraph();
27: }
```

شكل ٣-١٢تحديد كارت الشاشة والحالة المناسبة لها

وعند تنفيذ هذا البرنامج نحصل على النتيجة التالية.



شكل ٤-١٢رسم مجموعة مستطيلات

وعن هذا البرنامج نوضح ما يلي :-

يتم تحديد نوع كارت الشاشة مباشرة وذلك فى السطر رقم ٦ من البرنامج باستعمال الجملة

driver=DETECT

ومعنى كلمة DETECT أى أكتشف. وهى تحدد نوع الشاشة وتضع الرقم الدال على ذلك فى المتغير driver وكذلك تحدد أفضل حالة رسم لهذه الشاشة وتضع الرقم الدال عليها فى المتغير mode

فى السطر رقم ١٠ يتم استخدام القيمتين فى الدالة initgraph()

وبهذا الاسلوب فأن عملية التحويل الى حالة الرسم أصبحت سهلة جدا ولست مطالبا بمعرفة نوع الكارت المركب ونوع الشاشة.

وفى السطر رقم ١١ الدالة getmaxx() تحدد أقصى قيمة افقية ممكن الرسم فيها ، وهى القيمة العظمى للاحداثى الافقى للشاشة وتخزنها فى المتغير maxx

وفى السطر رقم ١٢ الدالة getmaxy() تحدد أقصى قيمة رأسية ممكن الرسم فيها وهى القيمة العظمى للاحداثى الرأسى للشاشة وتخزنها فى المتغير maxy وفى السطر رقم ١٨ استخدمنا دالة rectangle لرسم مستطيل. وتأخذ دالة رسم المستطيل أربعة معاملات هى Left , Top , Right , Bot حيث أن المتغيرات LEFT,TOP هما احداثى النقطة الاولى والمتغيران RIGHT,BOT هما احداثىالنقطة الثانية وهى متغيرات تم الاعلان عنها فى أول البرنامج.

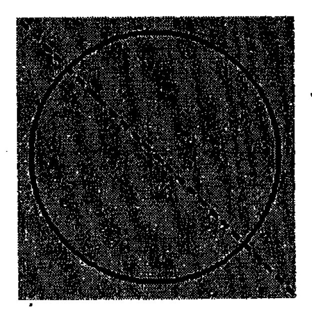
## رسم الخطوط و تغيير الالوان Lines and Colors

يمكن تغيير نوع خط الرسم وسمكه وذلك قبل استعمال دوال الرسم باستخدام الدالة setlinestyle() التى تقوم بتحديد نوع وسمك الخط ويتضح ذلك من البرنامج الموجود بالشكل رقم ٥-١٢ الذى يقوم برسم خط ودائرة بأشكال مختلفة ويظهر ذلك من نتيجة التنفيذ الموجودة بالشكل رقم ٦-١٢

```
0:   /*Program Name CS12_5.C*/
1:  #include <graphics.h>
2:  #define IGNORED 0
3:  void main (void)
4:    {
5:      int driver,mode;
6:      int x1=0,y1=0;
7:      int x2=199,y2=199;
8:      int xc=100,yc=100;
9:      int radius=90;
10:     driver=DETECT;
11:     initgraph(&driver,&mode," ");
11:     setcolor(4);setbkcolor(7);cleardevica();
12:     setlinestyle(DASHED_LINE,IGNORED,THICK_WIDTH);
13:     setcolor(BLUE);
14:     line (x1,y1,x2,y2);
15:     circle(xc,yc,radius);
16:     getch();
17:     closegraph();
18:  }
```

شكل رقم ٥-١٢ برنامج رسم خط دوائرة

و عند تنفيذالبرنامج نحصل على النتيجة التالية.



شكل رقم ٦-١٢رسم خط ودائرة

الجديد فى البرنامج الموجود بالشكل رقم ٥-١٢ أنه فى السطر رقم ١٢ تقوم الدالة setlinestyle() بتحديد مواصفات الخط حسب معاملات الدالة كما يلى :

*المعامل الاول* : وهو المتغير DASHED_LINE يحدد شكل الخط هل هو متواصل أم متقطع وهكذا،و هناك مجموعة أشكال للخطوط يمكن تحديد أحداها (تسمى line style) بالمتغير الدال على الشكل أو بالرقم المقابل له وهذه الاشكال هى

0 SOLID_LINE
1 DOTTED_LINE
2 CENTER_LINE
3 DASHED_LINE
4 USERBIT_LINE

*والمعامل الثانى* : هو المتغير IGNORED وقيمته صفر ومعناه تحديد نوع خط من الأنواع المعرفة باللغة.

*والمعامل الثالث:*: يمكن أن يأخذ احدى قيمتين هما :

1 NORM_WIDTH
3 THICK WIDTH

حيث ١ هو سمك الخط العادى أما ٣ فهو ثلاث أضعاف الخط العادى وقد استخدمنا فى هذا المثال THICK-WIDTH.

وفى السطر رقم ١٣ الدالة setcolor() لتغير اللون فى حالة الرسم وتشتمل على الرقم الدال على اللون أو اسم اللون

وفىالسطرين رقم ١٤ و ١٥ نستخدم دوال رسم الخط والدائرة .

## القطع الناقص والشكل المتعرج Ellipses and Polygons

نستخدم الدالة ellipse() لرسم قطع ناقص وهو الشكل البيضاوى وتأخذ الصورة التالية:

```
ellipse(xe,ye,stangle,endangle,xrad,yrad);
```

وفيها المتغيرات

xe,ye هما مركز القطع الناقص والمتغيران stangle,endangle هما زاوية البداية والنهاية

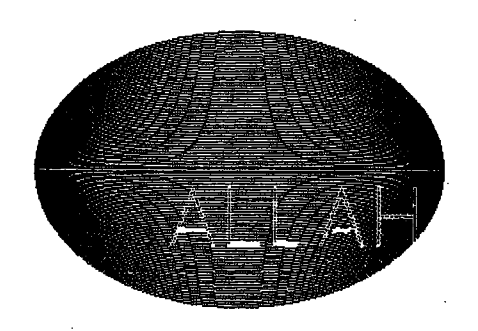
والمتغيران xrad,yrad هما نصف القطر (الافقى)في اتجاه x ونصف القطر الرأسى

والبرنامج الموجود بالشكل رقم ٧-١٢ يستخدم الدالة ()ellipse مع الدوارة for ليعطى نتيجة التنفيذ الموجودة بالشكل رقم ٨-١٢

```
0:  /*Program Name CS12_7.C*/
1:  #include <graphics.h>
2:  void main (void)
3:   {
4:      int driver,mode;
5:      int xe=250,ye=100;
6:      int xrad=150,yrad;
7:      int stangle=0,endangle=360;
8:      driver=DETECT;
9:      initgraph(&driver,&mode," ");
10:     for (yrad=0;yrad<100;yrad+=2)
11:             ellipse(xe,ye,stangle,endangle,xrad,yrad);
12:     moveto(200,90);
13:     setcolor(BLACK);
14:     settextstyle(3,0,7);
15:     outtext("ALLAH");
16:     getch();
17:     closegraph();
18:  }
```

شكل رقم ٨-١٢

وعند تنفيذ هذا البرنامج نحصل على النتيجة التالية :




شكل رقم ٩-١٢

وعن هذا البرنامج نوضح ما يلى:

فى السطر رقم ١٠ يتم إستعمال الدوارة for لتغير قيمة المتغير yrad من صفر إلى ١٠٠.

فى السطر رقم ١١ يتم استخدام الداله () ellipse مع تغير قيمه المتغير yrad وبالتالى يتم رسم أشكال مختلفه من القطع الناقص مشتركة فى المركز ومختلفة فى نصف القطر كما فى الشكل ٩-١٢.

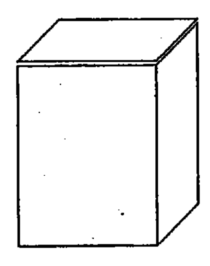
*رسم شكل غير منتظم (متعرج)*

الشكل المتعرج هو أى شكل يتكون من مجموعة خطوط تعطى شكل غير منتظم وتقوم الدالة ()drawpoly برسم الشكل المتعرج وتأخذ الصورة العامة الأتية ;drawpoly(pno,rightpara). والمعامل الاول للدالة هوالمتغير pno ويمثل عدد النقط المطلوب التوصيل بينها. و المعامل الثانى عبارة عن مصفوفة من الارقام تمثل النقط المراد تمثيلها ويستخدم البرنامج الموجود بالشكل رقم ١٠-١٢ الدالة ()drawpoly لرسم شكل متعرج كما يتضح من نتيجه التنفيذ الموجودة بالشكل رقم ١١-١٢

```
0:  /*Program Name CS12_10.C*/
1:  #include <graphics.h>
2:  #define LEFT 50
3:  #define TOP 50
4:  #define RIGHT 150
5:  #define BOT 180
6:  int rightpara[]={150,50, 180,20, 180,135, 150,180};
7:  int toppara[]={50,47 ,150,47, 180,17,95,17, 50,47};
8:  void main (void)
9:      {
10:         int driver,mode;
11:         driver=DETECT;
12:         initgraph(&driver,&mode," ");
13:         rectangle(LEFT,TOP,RIGHT,BOT);
14:         drawpoly(4,rightpara);
15:         drawpoly(5,toppara);
16:         getch();
17:         closegraph();
18:     }
```

الشكل رقم ١٠-١٢ برنامج رسم شكل متعرج

وعند تنفيذ البرنامج نحصل على النتيجة التالية:



الشكل رقم ١١-١٢رسم خط متعرج

وعن هذا البرنامج نوضح مايلي:

فى السطر رقم ٦ اعلان عن مصفوفة قيم صحيحة وتخزين هذه القيم بها ،وهذه القيم هى نقط متعدد الخطوط المطلوب التوصيل بينهـا ( نقـط الشـكل المتعـرج الايمـن كمـا فـى الشكل ١١-١٢) وبالمثل السطر رقم ٧ نقط متعدد الخطوط الافقى والسطر رقـم ١٣ يرسم مستطيل وهو الوجه الامامى للشكل والسطر رقم ١٤و١٥ يرسـم متعـدد الخطـوط الجانبى والعلوى حتى نحصل الشكل رقم ١١-١٢

## التلوين و التظليل و الاشكال المختلفة للتظليل

يمكـن تظليـل أى مسـاحة مغلقـة بأشـكال مختلفـة مـن أشـكال التظليـل وذلـك بتحديـد مواصفات شكل التظليل باستخدام الدالة setfillstyle() التى تأخذ الصورة التالية

**setfillstyle(PATTERNS,color)**

حيث المتغير PATTERNS هو شكل التظليل الذى يتم تحديده بالمتغير الدال عليه.

وفيما يلى قائمة المتغيرات المستخدمه فى تحديد شكل التظليل.

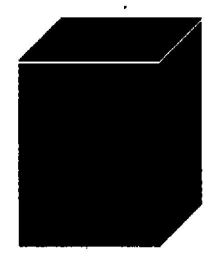
| | |
|---|---|
| EMPTY_FILL | uses background color |
| SOLID_FILL | uses solid fill color |
| LINE_FILL | --- fill |
| LTSLASH_FILL | /// fill |
| SLASH_FILL | /// fill thick lines |
| BKSLASH_FILL | /// fill thick lines |
| LTBKSLASH_FILL | /// fill |
| HATCH_FILL | light hatch |
| XHATCH_FILL | heavy cross hatch |
| INTERLEAVE_FILL | interleaving line |
| WIDE_DOT_FILL | widely spaced dots |
| CLOSE_DOT_FILL | closely spaced dots |
| USER_FILL | user-defined fill |

و المعامل الثانى للدالـه هـو لـون التظليـل (وتوجـد دالـة خاصـة برسـم متعـدد الخطـوط وتظليله فى نفس الوقت هى الداله fillpoly وهى تقـوم نفـس عمـل الدالـة drawpoly()

ولكن تظلل وتلـون الشـكل بالموصفـات المحـددة بالدالـة ()setfillstyle ) ، والبرنـامج الموجود بالشكل رقم ١٢-١٢ يرسم شكل متعرج مظلل الاوجه كما يتضـح مـن نتيجـة التنفيذ الموجودة بالشكل رقم ١٣-١٢

```
0:  /*Program Name  CS12_12.C*/
1:  #include <graphics.h>
2:  int rect[]={50,50 ,150,50, 150,180, 50,180, 50,50};
3:  int sidepara[]={150,50, 180,20, 180,135,150,180};
4:  int toppara[]={50,47, 150,47, 180,17, 95,17, 50,47};
5:  void main (void)
6:    {
7:          int driver,mode;
8:          driver=DETECT;
9:          initgraph(&driver,&mode," ");
10:         setfillstyle(11,GREEN);
11:         fillpoly(5,rect);
12:         setfillstyle(1,RED);
13:         fillpoly(4,sidepara);
14:         fillpoly(5,toppara);
15:         getch();
16:         closegraph();
17:     }
```

شكل رقم ١٢-١٢رسم شكل متعرج مظلل



شكل رقم ١٣-١٢ شكل متعرج مظلل

فى البرنامج الموجود بالشكل رقم ١٢-١٢ فى السطر رقم ١٠ استخدمنا الدالة ()setfillstyle لتحديد شكل ولون التظليل و فى السطر رقم ١١ استخدمنا الدالة ()fillpoly لرسم شكل متعدد خطوط مظلل.

## الرسم البيانى ذو البعدين وثلاثى الأبعاد [2 -DIM ,3-DIM]

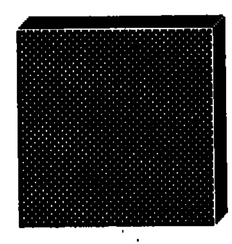
من الدوال المفيدة فى الرسم البيانى الدالتين ()bar(),bar3d حيث تقوم الدالة ()bar برسم مستطيل وتأخذ الصورة bar(l,t,r,b) ومعاملاتها l,t,r,b هى left,top,right,bottom كما فى دالة رسم المستطيل والفرق بينها وبين دالة رسم المستطيل أن الدالة () bar ترسم مستطيل مظلل بشكل التظليل الذى تم تحديده مسبقا ، والدالة ()bar3d ترسم مستطيل ثلاثى الإبعاد وهو يمثل الرسم فى ٣ اتجهات وتأخذ الصورة bar3d(l,t,r,b,z,f)

وفيها المتغير z هو البعد الثالث والمتغير f يأخذ أحد قيمتين أما صفر أو ١ والقيمة صفر تعنى عدم رسم المستطيل العلوى للمكعب بينما القيمة ١ تعنى رسمه ، والبرنامج الموجود بالشكل رقم ١٤-١٢ يستخدم الدالة ()bar3d فى رسم مستطيل ثلاثى الابعاد كما يظهر من الشكل رقم ١٥-١٢

```
/*Program Name CS12_14.C*/
#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
int main(void)
{
  /* request auto detection */
  int gdriver = DETECT, gmode, errorcode;
  int midx, midy, i;
  /* initialize graphics, local variables */
  initgraph(&gdriver, &gmode, "");
```

```
/* read result of initialization */
errorcode = graphresult();
if (errorcode != grOk)  /* an error occurred */
 {
    printf("Graphics error: %s\n", grapherrormsg(errorcode));
    printf("Press any key to halt:");
    getch();
    exit(1); /* terminate with error code */
 }
midx = getmaxx() / 2;
midy = getmaxy() / 2;
/* loop through the fill patterns */
for (i=EMPTY_FILL; i<USER_FILL; i++)
  {
     /* set the fill style */
     setfillstyle(i, getmaxcolor());
     /* draw the 3-d bar */
     bar3d(midx-50, midy-50, midx+50, midy+50, 10, 1);
     getch();
  }
 /* clean up */
 closegraph();
 return 0;
}
```

شكل رقم ١٤-١٢برنامج رسم مستطيل ثلاثي الأبعاد



شكل رقم ١٥-١٢مستطيل ثلاثي الأبعاد

## رسم الشكل الدائرى Pie

يستخدم الشكل الدائرى لتمثيل مجموعة قيم فى شكل رسم بيانى ويتم رسم الشكل الدائرى باستخدام الدالة () pieslice التى تأخذ الصورة التالية

pieslice(X,Y,startangle,endangle,RADIUS);

حيث :

المتغيران X,Y هما احداثى نقطه مركز الشكل الدائرى ، والمتغير startangle هو زاوية بداية رسم الشكل الدائرى

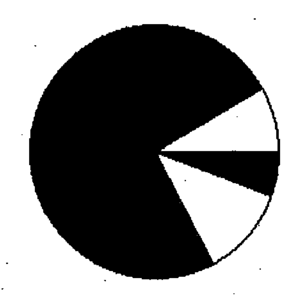
والمتغير endaangle هو زاوية النهاية والمتغير RADIUS هو نصف القطر

والبرنامج الموجود بالشكل رقم ١٦-١٢ يستخدم الدالة ()pieslice فى رسم شكل دائرى يمثل عدة قيم ويظهر ذلك فى نتيجه التنفيذ الموجودة بالشكل رقم ١٧-١٢

```
0:  /*Program Name CS12_16.C*/
1:  #include <graphics.h>
2:  #define N 6
3:  #define RADIUS 90
4:  #define X 100
5:  #define Y 100
6:  int data[N]={11,19,44,32,15,7,};
7:  void main(void)
8:     {
9:         float datasum,startangle,endangle,relangle;
10:        int driver,mode,j;
11:        driver=DETECT;
12:        initgraph(&driver,&mode," ");
13:        for (j=0,datasum=0;j<N;j++)
14:            datasum +=data[j];
15:        endangle=0;
16:        for (j=0;j<N;j++)
17:          {
```

```
18:            startangle=endangle;
19:            relangle=360*(data[j]/datasum);
20:            endangle=startangle+relangle;
21:            setfillstyle(SOLID_FILL, j % 4 );
22:            pieslice(X,Y,startangle,endangle,RADIUS);
23:        }
24:     getch();
25:     closegraph();
26:  }
```

شكل رقم ١٦-١٢ برنامج رسم شكل دائرى



شكل رقم ١٧-١٢ شكل دائرى

وعن هذا البرنامج نوضح ما يلى :

- فى السطر رقم ١٣ يتم استعمال دورة for للتعامل مع القيم المخزنه بالمصفوفة data [N].
- فى السطر رقم ١٦ يتم استعمال دورة for لتغيير مقدار زاوية البداية وزاوية النهاية للشكل الدائرى.
- فى السطرين ١٩ ، ٢٠ يتم حساب زاويتى البداية والنهاية للشكل الدائرى كل مرة.
- فى السطر رقم ٢٢ يتم رسم الشكل الدائرى باستعمال الداله pieslice () وذلك بالمعاملات التى تم تحديدها.

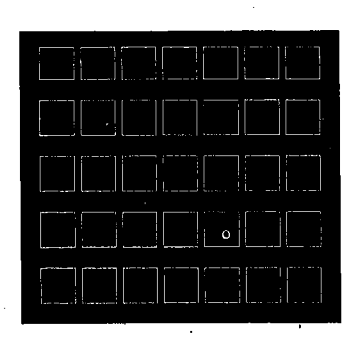
## رسم خطوط بالنسبة لاخر نقطة ( الرسم النسبى)

شرحنا فى أول الفصل الدالة line() التى ترسم خط وفيها لابد من تحديد نقطتى الخط ولكن فى بعض الاحيان نحتاج لرسم خط بمعلومية نقطة واحدة والنقطة الثانية هى النقطة الحالية أو النقطة التى يمكن تحديدها بالدالة moveto(x,y) ،و يتم تحقيق ذلك باستخدام الدالة linerel(x,y) التى ترسم خط من المكان الحالى للمؤشر الى النقطة المحددة ب x,y وهذا يفيد فى الرسومات البيانية، والبرنامج الموجود بالشكل رقم ١٨-١٢ يستخدام الدالة linerel(x,y) لرسم مجموعه مستطيلات كما يظهر ذلك فى الشكل رقم ١٩-١٢

```
/*Program Name cs12_18.c */
#include <graphics.h>
#define MAX 200
#define GRID 30
#define SIDE 25
void square(int side);
void main (void)
  {
int driver,mode;
int x,y;
driver=DETECT;
initgraph(&driver,&mode," ");
    for (y=0;y<MAX;y+=GRID+10)
     for(x=0;x<MAX;x+=GRID)
       {
          moveto(x,y);
          square(SIDE);
       }
getch();
closegraph();
}
```

```
void square(int side)
  {
      linerel(side,0);
      linerel(0,side);
      linerel(-side,0);
      linerel(0,-side);
  }
```

شكل رقم ١٨-١٢برنامج رسم مجموعة مستطيلات



شكل رقم ١٩-١٢مجموعة مستطيلات

وعن هذ البرنامج نوضح ما يلى :

فى البرنامج الموجود بالشكل رقم ١٨-١٢ فى السطر رقم ١٥ نستخدم الدالة moveto() لوضع المؤشر عند النقطة x,y (التى تتغيرباستخدام دوارتي for) وفى السطر رقم ١٦ نستدعى الدالة square() التى ترسم مربع باستخدام الدالة linerel()وفى السطر رقم ٢١ تبدأ الدالة () square باستقبال رقم ثابت هو المتغير SIDE وقيمته ٣٠ وتستخدمه الدالة lienrel() لرسم مربع طول ضلعه ٣٠ و بتغير موضع المؤشر بالدالة moveto() تحصل على شكل رقم ١٩-١٢ وهو شبكة من المربعات.

## رسم النقط على الشاشة (Pixels)

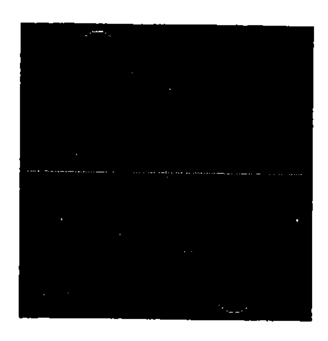
للنقطة (pixle) استخدمات كثيرة منها رسم أى معادلة بالنقط المتقاربة، ويتم رسم نقطة باستخدام الدالة ;putpixel(x,y,color) حيث المتغيران x,y هما احداثى النقطة والمتغير color هو لون النقطة المطلوب رسمها ، ومن التطبيقات التى تستخدم النقطة رسم المعادلات الرياضية

والبرنامج الموجود بالشكل رقم ٢٠-١٢ يستخدم الدالة ()putpixle فى رسم الدالة الجيبية ( sine wave )كما يتضح من نتيجة التنفيذ الموجودة بالشكل رقم ٢١-١٢ .

```
/* Program Name cs12_20.c */
#include <dos.h>
#include <graphics.h>
#include <math.h>
void main (void)
   {
      int driver,mode;
      double angle,sineofa;
      int x,y;
      driver=DETECT;
      initgraph(&driver,&mode," ");
      line (0,100,200,100);
      for (x=0;x<200;x++)
         {
            angle=( (double)x/200 *(2*3.14159265));
            sineofa=sin(angle);
            y=100-100*sineofa;
            putpixel(x,y,WHITE);
            sound(50*x);
            delay(3);
         }
      nosound();
      getch();
```

```
    closegraph();
}
```

شكل رقم ٢٠-١٢ برنامج رسم النقط



شكل رقم ٢١-١٢ رسم النقط

## تحريك الرسومات Animations

من العمليات المشهورة فى الرسم تحريك الرسومات ويتم ذلك بأكثر من طريقة من هذه الطرق تخزين الشكل المراد تحريكه فى متغير مع مسح الشكل من الشاشه ثم رسم الشكل فى موضع آخر ثم مسح الشكل وهكذا فيظهر وكأنه يتحرك ويتم ذلك بإستخدام الدالتين getimage() putimage() والدالة getimage() تأخذ الصورة العامة التاليه :

**getimage(x1,y1,x2,y2,clip)**

حيث المتغيرات x1,y1,x2,y2 هى احداثيات المساحة المطلوب تخزينها.

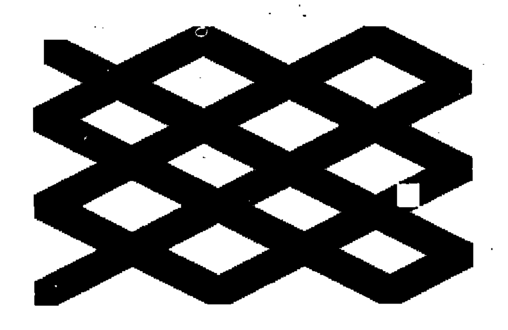
والمتغير Clip يأخذ القيمة صفر أو القيمه 1 ، و القيمه 1 معناها مسح الشكل بعد تخزينه ، والقيمة صفر معناها عدم مسح الشكل بعد تخزينه.

والبرنامج الموجود بالشكل رقم ٢٢-١٢ يقوم برسم دائرة وتحريكها على الشاشة كما يتضح من الشكل رقم ٢٣-١٢

```
/* Program Name cs12_22.c */
#include <stdlib.h>
#include <conio.h>
#include <dos.h>
#include<graphics.h>
#define left 0
#define top 0
#define right 319
#define bottom 199
#define radius 8
void main (void)
 {
      int driver,mode;
      int x,y,dx,dy,oldx,oldy;
      void *ballbuff;
      unsigned size;
      driver=DETECT;
      initgraph (&driver,&mode," ");
      rectangle(left,top,right,bottom);
      x=y=radius+10;
      setcolor(WHITE);
      setfillstyle(1,WHITE);
      circle(x,y,radius);
      floodfill(x,y,GREEN);
      size=imagesize(x-radius,y-radius,x+radius,y+radius);
      ballbuff=(void *)malloc(size);
      getimage(x-radius,y-radius,x+radius,y+radius,ballbuff);
      dx=2;
      dy=1;
     while ( !kbhit() )
        {
           putimage(x-radius,y-radius,ballbuff,COPY_PUT);
           oldx=x,oldy=y;
           x+=dx,y+=dy;
```

```
            sound(10*x);
            delay(10);
            if (x<=left+radius+2 || x>=right-radius-2)
                dx=-dx;
            if (y<=top+radius+1 || y>=bottom-radius-1)
            dy=-dy;
            putimage(oldx-radius,oldy-radius,ballbuff,XOR_PUT);
        }
    nosound();
    closegraph();
}
```

شكل رقم ٢٢-١٢ برنامج رسم دائرة وتحريكها



شكل رقم ٢٣-١٢ تحريك الدائره

## استخدام خطوط الكتابه

يمكن فى حالة الرسم كتابة الكلمات بفونتات وأحجام مختلفة مما يعطى مخرجات جذابة – وذلك بتحديد مواصفات الكتابة مثل اسم الفونت والحجم واتجاه الكتابة ويتم ذلك باستخدام الدالة settextstyle( ) ثم تحديد مكان الكتابة على الشاشه باستخدام الدالة moveto( ) ثم طباعة الكلمات المراد طباعتها باستخدام الدالة outtext( ).

ويمكن الكتابة فى موضع معين مباشرة باستعمال الدالة outtextxy() التى تطبع الكلمات عند نقطة محددة (X,Y)

والدالة settextstyle( ) تأخذ الصورة التالية:-

settextstyle(FONT ,DIRECTION ,FONT_SIZE);

حيث :

المتغير FONT: هو الاسم أو الرقم الدال على الفونت المطلوب الكتابة به ويتم اختيــاره من الجدول التالى وذلك بكتابة المتغير أو الرقم الدال عليه

| | |
|---|---|
| 0 | DEFAULT_FONT |
| 1 | TRIPLEX_FONT |
| 2 | SMALL_FONT |
| 3 | SANS_SERIF_FONT |
| 4 | GOTHIC_FONT |

المتغيرDIRECTION : يحدد اتجاه الكتابة ويكون أحد الاختيارين التاليين :

| | | | |
|---|---|---|---|
| 0 | HORIZ_DIR | left to right | افقى |
| 1 | VERT_DIR | bottom to top | راسى |

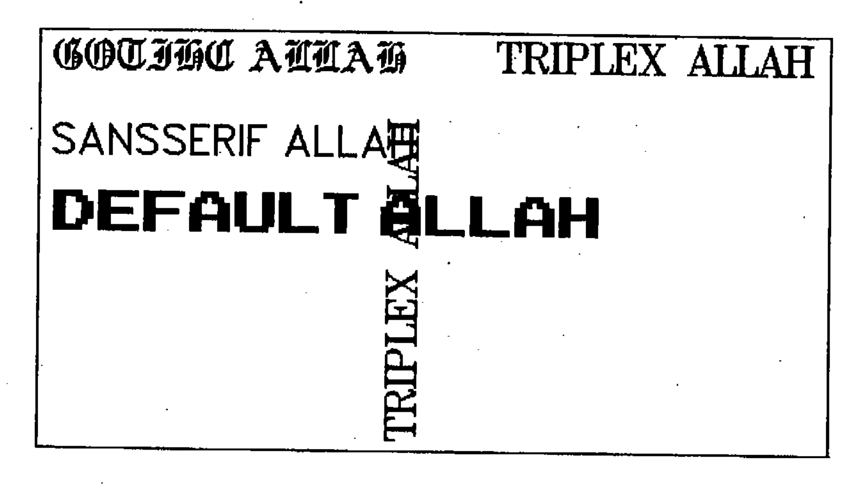
المتغير FONT_SIZE : يحدد حجم الفونت ويأخذ أرقام تعبر عن حجم الفونت .

يقوم البرنامج الموجود بالشكل رقم ٢٤-١٢ بكتابة مجموعة كلمات بفونتـات مختلفـة وأحجام مختلفة ويتضح ذلك من نتيجة التنفيذ الموجوده بشكل رقم ٢٥-١٢

```
0:  /* program name cs12_24.c */
1:  #include <graphics.h>
2:  #define FONT_SIZE 4
3:  void main (void)
4:      {
5:        int driver,mode;
6:        driver=DETECT;
7:        initgraph(&driver,&mode," ");
8:        settextstyle(4,0,FONT_SIZE);
```

```
9:      outtext("GOTIHC  ALLAH   ");
10:     settextstyle(1,0,FONT_SIZE);
11:     outtext("TRIPLEX ALLAH   ");
12:     moveto(0,60);
13:     settextstyle(3,0,FONT_SIZE);
14:     outtext("SANSSERIF  ALLAH   ");
15:     moveto(0,120);
16:     settextstyle(0,0,FONT_SIZE);
17:     outtext("DEFAULT ALLAH    ");
18:     moveto(250,0);
19:     settextstyle(1,1,FONT_SIZE);
20:     outtext("TRIPLEX ALLAH    ");
21:      getch();
22:     closegraph();
23:   }
```

شكل رقم ٢٤-١٢ برنامج استخدام خطوط مختلفة للكتابة




شكل رقم ٢٥-١٢ الكتابة بخطوط مختلفة

**وعن هذا البرنامج نوضح ما يلى :**

في السطر رقم ٨ تم تحديد مواصفـات الكتابـة بالدالـة settextstyle() وفيهـا المعامل الأول هو الرقم ٤ الذى يحـدد نـوع الفونـت المطلوب الكتابـة بـه ويؤخـذ مـن الجدول السابق. وفى هذا المثال اخترنا الفونت GOTHIC_FONT ، والمعـامل الثـانى يحدد اتجاه الكتابة وفى هذا البرنامج اخترنا الاتجاه الأفقـى والمعـامل الثـالث هـو حجـم الفونـت المطلـوب الكتابـة بـه ،وفـى السـطر رقـم ٩ تـم طباعـة الكلمـات على الشاشـة باستخدام الدالة outtext() وهكذا باقى البرنامج .

*مثال:*

لا يعتمد الرسم الجيد فى البرمجة على استخدام دوال الرسم المتوفرة فقـط بـل يعتمـد أكثر على الافكار الجيدة لاستغلال هـذه الـدوال ولتوضيـح هـذه الفكـرة نتـابع البرنـامج التالى فلا نجد به دوال جديدة ولكن الجديد فيـه هـو فكـرة اظهـار خطـوط الكتابـة علـى شكل رسم حيث يقوم باستعمال دوال الكتابة بفونتات مع دوارة for لطباعة كلمة علـى الشاشة هذه الكلمة تطبع ٢٠ طبقة فوق بعضها بالوان مختلفـة ممـا يعطى شـكل جـذاب كما فى الشكل ٢٦-١٢

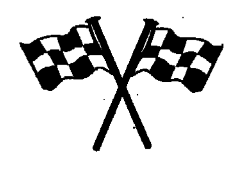
```
0:  /* program name cs12_26.c */
1:  #include <graphics.h>
2:  #define FONT_SIZE 6
3:  void main (void)
4:    {
5:       int driver,mode,i;
6:       driver=DETECT;
7:       initgraph(&driver,&mode," ");
8:       settextstyle(1,0,FONT_SIZE);
9:       for (i=0;i<20;i++)
10:          {
11:             setcolor (i);
12:             outtextxy(50+i,(getmaxx()/2)+i, "ENG_AZAB .. ");
13:             delay(200);
```

```
14:         }
15 :    getch();
16:     closegraph();
17:  }
```

الشكل رقم ٢٦-١٢ برنامج طباعة الابناط بألوان مختلفة

ENG.A Z A B

شكل ٢٧-١٢ الكتابة بطبقات متعددة



# الفصل الثالث عشر

# دوال التعامل مع الذاكرة

# Memory Allocation

في هذا الفصل نتناول موضوع التعامل مع الذاكرة مثل حجز أماكن في الذاكرة حسب الطلب والغاء حجز أماكن المتغيرات بعد الانتهاء من استعمالها وذلك من خلال مجموعة دوال هي :

- الدالة () malloc
- الدالة () free
- الدالة () realloc
- الدالة () calloc
- الدالة () faralloc

شرحنا فى فصل المصفوفات كيفية الاعلان عن مصفوفة. فمثلاً مصفوفة عدد عناصرها ١٠٠ عنصر تتطلب من البرنامج حجز ١٠٠ مكان فى الذاكرة وعادة لا نستعمل جميع عناصر المصفوفة وبالتالى تبقى بعض عناصر المصفوفة غير مستعملة وهذا استعمال سيىء للذاكرة

وأيضا بعد حجز متغيرات والانتهاء من استعمالها تظل الاماكن محجوزة فى الذاكرة وهذا يشغل حيز من الذاكرة لذلك يفضل مسحها ، وقبل أن نشرح دوال التعامل مع الذاكرة نراجع كيفية التعامل مع المصفوفة بالطريقة المعتادة

البرنامج الموجود فى الشكل رقم ١-١٣ يقوم بالاعلان عن مصفوفة حرفيات عدد عناصرها ١٠٠ عنصر فى حين أنك قد لا تستخدم كل هذه العناصر ولأننا لا نعرف العدد المراد حجزه نقوم بحجز عدد كبير وليكن ١٠٠ عنصر يستخدم منهم البرنامج ما يشاء والباقى يعتبر مساحة غير مستغلة من الذاكرة

```
0:   /*Program Name CS13_1.C*/
1:   #include <stdio.h>
2:   #define MAX 100
3:   void main(void)
4:       {
5:           char stud_name[MAX][35];
6:           long x;
7:           printf("\nEnter student names, a blank line will end\n");
8:           for ( x = 0; x < MAX; x++ )
9:               {
10:                  printf("Enter student  %5d: ", x+1);
11:                  gets(stud_name[x]);
12:                  if( stud_name[x][0] == '\0')
13:                  x = MAX;
14:              }
15:           printf("\n\nYou entered :\n");
16:           for ( x = 0; stud_name[x][0] != '\0' && x < MAX; x++ )
```

```
17:          {
18:            printf("\nStudent %5d:", x+1);
19:            printf(" %s", stud_name[x]);
20:          }
21:    }
```

الشكل رقم ١-١٣ برنامج استخدام مصفوفة محددة العناصر

وعند تنفيذ هذا البرنامج نحصل على النتيجة التالية :

```
Enter student names, a blank line will end
Enter student 00001: samy mohamed azab
Enter student 00002: hamdy mohamed azab
Enter student 00003: nabil mohamed azab
Enter student 00004: azab mohamed azab
Enter student 00005:

You entered :
Student 00001: samy mohamed azab
Student 00002: hamdy mohamed azab
Student 00003: nabil mohamed azab
Student 00004: azab mohamed azab
```

الشكل رقم ٢-١٣ نتيجة التنفيذ

وعن هذا البرنامج نوضح ما يلى :

- يقوم السطر رقم ٥ بالاعلان عن مصفوفة من نوع حرفى عدد عناصرها ١٠٠ لاتزيد عدد حروف كل منها عن ٣٥ حرفاً
- وفى السطر رقم ٨ الدوارة for للتكرار ١٠٠ مرة.
- وفى السطر رقم ١١ الدالة gets() تستقبل متغير حرفى يمثل اسم الطالب وتخزنه فى عنصر المصفوفة المقابل.

- وفي السطر رقم ١٢ شرط أذا كانت القيمة المدخلة صفر أى أن المستخدم ضغط علي المفتاح Enter بدون ادخال قيمة يأخذ المتغير x القيمة max وبالتالى ينتهى التكرار ويتوقف ادخال البيانات
- وفي السطر رقم ١٦ تبدأ دوارة for لطباعة البيانات التى أدخلها المستخدم

## ملاحظات على البرنامج

فى السطر رقم ٥ تم الاعلان عن مصفوفة عدد عناصرها ١٠٠ عنصر فى حين أن العناصر المستخدمة من المؤكد أنها لا تصل الى ١٠٠ عنصر. اذا باقى العناصر الغير مستخدمة تعتبر محجوزة فى الذاكرة بدون استعمال وهذا استعمال سيىء للذاكرة لذلك تقوم دوال التعامل مع الذاكرة بتحديد عدد الاماكن المطلوبة فقط وهذا ما نراه فى البرنامج الموجود بالشكل رقم ٣-١٣ الذى يسأل المستخدم عن عدد الطلاب المراد ادخال بيانتهم ثم يقوم بحجز اماكن بهذا العدد فقط وقبل أن نتابع البرنامج نشرح الدالة المستخدمة فى حجز أماكن حسب الطلب.

## الدالة malloc()

تقوم الدالة بحجز عدد من الاماكن فى الذاكرة وتأخذ الصورة التالية :

```
void ptr = *malloc(size )
```

ومعناها احجز عدد من الأماكن مقداره size وضع عنوان هذه الأماكن فى المؤشر ptr

والدالة تعيد مؤشر الى هذه الأماكن التى تم حجزها فإذا لم تتمكن الدالة من حجز أماكن لسبب ما (أن الذاكرة غير كافية مثلا) تعيد القيمة NULL (\0) ، و للتأكد من حجز الاماكن نختبر قيمة المؤشر. ويتضح ذلك من البرنامج الموجود بالشكل رقم ٣-١٣

```
0:    /*Program Name CS13_3.C*/
1:    #include <stdio.h>
```

```
2:   #include <stdlib.h>
3:   void main(void)
4:     {
5:        long nbr_students = 0;
6:        long ctr;
7:        char *stud_name;
8:        char trash[80];    /* to clear keyboard buffer */
9:        while( nbr_students < 1 || nbr_students > 2000000000 )
9:           {
10:             printf("\nHow many students will be entered? ==> ");
11:             scanf("%d", &nbr_students);
12:             gets(trash);     /* clear out keyboard buffer */
13:           }
14:       stud_name = (char *) malloc( 35*nbr_students);
15:       if( stud_name == NULL )   /* verify malloc() was successful */
16:           {
17:              printf( "\nError in line %3.3d: Could not allocate
                                memory.",__LINE__); exit(1);
18:           }
19:       for( ctr = 0; ctr < nbr_students; ctr++ )
20:           {
21:               printf("\nEnter student %5d: ", ctr+1);
22:               gets(stud_name+(ctr*35));
23:           }
24:       printf("\n\nYou entered the following:\n");
25:       for ( ctr = 0; ctr < nbr_students; ctr++ )
26:          {
27:             printf("\nStudent %5d:", ctr+1);
28:             printf(" %s", stud_name+(ctr*35));
29:          }
30:    }
```

شكل ٣-١٣ استخدام الدالة malloc() لتحديد عدد العناصر المطلوبة

وعند تنفيذالبرنامج نحصل على النتيجة التالية :

```
How many students will be entered? ==> 5
Enter student 00001: samy
Enter student 00002: hamdy
Enter student 00003: nabil
Enter student 00004: mohamed
Enter student 00005: azab
You entered the following:
Student 00001: samy
Student 00002: hamdy
Student 00003: nabil
Student 00004: mohamed
Student 00005: azab
```

شكل ٤-١٠ نتيجة التنفيذ

وعن هذا البرنامج نوضح ما يلي :

- يقوم السطر رقم ١٥ بطباعة رسالة الى المستخدم تسأله عن عدد الطلاب.
- السطر رقم ١١ يستقبل البرنامج رقم صحيح وهو عدد الطلاب التى يريد المستخدم أن يدخلها.
- وفى السطر رقم ١٤ الدالة malloc() تحجز عدد من الاماكن مقدارها nbr_students العدد الذى أدخله المستخدم مضروبا فى ٣٥ وهو عدد حروف كل عنصر لان الدالة malloc() تحجز بالبايت وبالتالى تحجز ١٤ عنصر طول كل عنصر ٣٥ حرف وتجعل المؤشر stud_name يشير الى هذه الاماكن

  والجزء (char *) يسمى Type Casting (راجع الفصل التاسع)
- وفى السطر رقم ١٥ جملة If تختبر قيمة المؤشر الى الحرفى stud_name فإذا كان يساوى null فمعنى هذا ان الدالة malloc() لم تتمكن من حجز

أماكن لأن الذاكرة غير كافية وبالتالى يطبع السطر رقم ١٧ رسالة تفيد بذلك وفيه الدالة exit() التى تنهى البرنامج

- فى السطر رقم ١٩ نستعمل دوارة for بعدد المرات التى حددها المستخدم لإستقبال أسماء الطلبة

وفىالسطر رقم ٢٢ الدالة gets() تستقبل هذه العناصر التى تمثل أسماء طلبة و نلاحظ هنا أن الزيادة تتم على العنوان فمثلا المتغير stud_name+(ctr*35) من أول قيمة المتغير ctr تساوى صفر وبالتالى المعادلة تعطى stud_name وهو عنوان أول عنصر ، وثانى قيمة للمتغير ctr هى ١ وبالتالى يزداد العنوان بمقدار ٣٥ بايت وهكذا...

- وفى السطر رقم ٢٥ تبدأ دوارة for أخرى لطباعة البيانات التى ادخلت والملاحظ هنا أن الدالة malloc() ساعدتنا على حجز الاماكن المطلوبة فقط وبالتالى ليس هناك أماكن محجوزة وغير مستعملة.

## الدالة realloc()

تقوم الدالة بتغيير عدد الاماكن التى تم حجزها وبالتالى لانحتاج لتحديد عدد الاماكن من البداية وتأخذالصورة التالية :

```
ptr=realloc(size)
```

حيث أن المتغير size هو الحجم الجديد الذى نريد حجزه

نفرض أنك تريد ادخال مجموعة من القيم ولكن لاتعرف عدد هذه القيم ولاتريد تحديده ، بل تريد أن تحجز مساحة عنصر واحد وكلما أدخلت عنصر حجز البرنامج له مكان فى الذاكرة .لتحقيق ذلك نستعمل الدالة realloc() بحيث نحجز مساحة عنصر واحد ثم نزيد المساحة كلما أراد المستخدم أن يدخل بيانات جديدة

وهذا نراه من فى البرنـامج الموجـود فـى الشـكل رقـم ٥ -١٣ وهـذا البرنـامج لايسـأل المستخدم على عدد الطلاب كما فى البرنامج الموجود بالشكل رقم ٣-١٣ولكن يسأل مباشرة عن اسم الطالب ويظل يستقبل بيانات الطلبة حتى يضغط المستخدم مفتـاح الادخال بدون بيانات

```
0:   /*Program Name CS13_5.C*/
1:   #include <stdio.h>
2:   #include <stdlib.h>
3:   #define NAME  35
4:   void main(void)
5:      {
6:         long student_cont   = 0;
7:         long ctr;
8:         char *stud_name = NULL;
9:         while((stud_name =
           realloc( stud_name, (NAME * (student_cont+1))))!= NULL )
10:            {
11:               printf("\nEnter student %5.#d: ", student_count+1);
12:               gets(stud_name+( student_cont   * NAME));
13:               If( stud_name[student_cont   * NAME] == NULL )
14:                  {
15:                    break;
16:                  }
17:                else
18:                  {
19:                    student_cont++;
20:                  }
21:            }
22:         printf("\n\nYou entered the following:\n");
23:         for ( ctr = 0; ctr < student_cont; ctr++ )
24:            {
25:                printf("\nStudent %5. d:", ctr+1);
```

```
26:             printf(" %s", stud_name+(ctr*NAME));
27:         }
28:     free(stud_name);
29: }
```

شكل ٥-١٣ استخدام الداله realloc()

عند تنفيذ : البرنامج نحصل على النتيجة التالية :

```
Enter student 00001: samy mohamed azab
Enter student 00002: hamdy mohamed azab
Enter student 00003: nabil mohamed azab
Enter student 00004: azab mohamed azab
Enter student 00005:
You entered the following:
Student 00001: samy mohamed azab
Student 00002: hamdy mohamed azab
Student 00003: nabil mohamed azab
Student 00004: azab mohamed azab
```

شكل ٦-١٣ نتيجة تنفيذ برنامج cs3_5.c

وعن هذا البرنامج توضح ما يلى :

- فى السطر رقم ٩ يقوم البرنامج بحجز مكان لعنصر واحد حيث قامت الدالة realloc() أول مرة بحجز مساحة فى الذاكرة مقدارها NAME أى ٣٥ بايت
- وفى السطر رقم ١٢ تستقبل الدالة gets() اسم طالب وتخزنه فى المكان الذى يشير اليه المؤشر stud_name +(student_cont*NAME) وفيه أول قيمة للمتغير student_count تساوى صفر وبالتالى تخزين أول اسم فى أول عنوان ، ثم يزداد هذا العنوان كلما أدخل المستخدم اسم آخر وذلك نتيجة

زيادة قيمة المتغير student_cont وبالتالي تزداد المساحة المحجوزة ، من هذا المثال نجد أن الدالة ()realloc ساعدتنا على حجز أماكن حسب الطلب فقط وبدون تحديد عدد الاماكن مسبقا مما يحقق أفضل استخدام للذاكرة.

## الدالة free ( )

تقوم هذه الدالة باخلاء الذاكرة من القيم التى تم استعمالها والتى لم يعد هناك حاجة الى وجودها فى الذاكرة وتأخذ الدالة الصورة التاليه

free (ptr)

ومعناها أجعل المؤشر ptr حر بحيث لايشير إلى أماكن بالذاكرة وبالتالى تصبح هذه الاماكن جاهزة للاستعمال من قبل أى برنامج آخر راجع البرنامج الموجود بالشكل رقم ٥-١٣ والبرنامج الموجود بالشكل رقم ٧-١٣للتعرف على كيفية استخدام الدالة (free) فى الغاء حجز المتغيرات التى انتهى العمل بها

## الدالة calloc()

تقوم الدالة بحجز أماكن فى الذاكرة بعدد معين وتأخذ الصورة التالية :

ptr=calloc(no,size)

والدالة تقوم بنفس عمل الدالة ()malloc ولكن تختلف فى طريقة الاستعمال حيث تقوم بحجز عدد من الاماكن هذا العدد هو no وحجم كل مكان هو size.

والبرنامج الموجود بالشكل رقم ٧-١٣ يوضح استعمال الدالة ()calloc

```
0:  /*Program Name CS13_7.C*/
1:  #include <stdio.h>
2:  #include <stdlib.h>
3:  void main( void )
4:    {
5:      int  nof_grades = 0;
6:      int  total = 0;
```

```
7:      int  ctr;
8:      int  * stud_grades   ;
9:     char trash[80];           /* to clear keyboard buffer */
10:    while( nof_grades  < 1||nof_grades >= 10000 )
11:      {
12:         printf("\nHow many grades will be entered? ==> ");
13:        scanf("%ld", &nof_grades);
14:        gets(trash);          /* clear out keyboard buffer */
15:      }
16:    stud_grades  = (int *) calloc( nof_grades   , sizeof(int));
17:    if(stud_grades == NULL )
18:      {
19:         printf( "\nError in line %3.3d : Could not allocate memory"
                                                          , __LINE__);
20:         exit(1);
21:      }
22:    for( ctr = 0; ctr < nof_grades   ; ctr++ )
23:      {
24:         printf("\nEnter grade %4.4d: ", ctr+1);
25:         scanf("%d",  stud_grades   +ctr);
26:      }
27:    printf("\n\nYou entered the following:\n");
28:    for ( ctr = 0; ctr < nof_grades   ; ctr++ )
29:      {
30:        printf("\nGrade %4.4d:", ctr+1);
31:        printf(" %d", *( stud_grades   +ctr));
32:        total += *( stud_grades   +ctr);
33:      }
34:    printf("\n\nThe average grade is: %d\n\n", (total/nbr_grades));
35:    free( stud_grades   );
36:  }
```

شكل ٧-١٣ استخدام الدالة calloc()

وعند تنفيذ : البرنامج نحصل على النتيجة التالية

```
How many grades will be entered? ==> 3
Enter grade 0001: 1000
Enter grade 0002: 789
Enter grade 0003: 5687
You entered the following:
Grade 0001: 1000
Grade 0002: 789
Grade 0003: 5687
The average grade is: 2492
```

شكل ٨-١٣ نتيجة تنفيذ برنامج csl3-7.c

وعن هذا البرنامج نوضح ما يلى :

فى السطر رقم ١٦ يتم استخدام الدالة ()calloc فى حجز مساحة بعدد العناصر المطلوبة وهى كما ترى تشبه بدرجة كبيرة الدالة ()malloc لا أن قاعدة الحجز تختلف حيث أن الدالة ()malloc تحجز مساحة مقدارها عدد البايت المكتوب بين القوسين malloc(50).

ولكن الدالة ()calloc تأخذ الصورة (calloc(5,10 ومعناها أحجز ٥ أماكن مساحة المكان الواحد ١٠ بايت وباقى البرنامج كما فى البرنامج الموجود بالشكل رقم ٣-١٣

## الدالة ()faralloc

تقوم الدالة بحجز أماكن فى الذاكرة وتختلف هذه الدالة فى أن الاماكن التى تحجزها يمكن أن تتعدى ٦٤ كيلو بايت وتأخذ الصورة التالية :

```
ptr= faralloc(size)
```

ويقوم البرنامج الموجود فى الشكل رقم ٩-١٠ باستعمال الدالة ()faralloc فى حجز أماكن فى الذاكرة تتعدى مساحتها ٦٤ كيلو بايت

```
0:    /*Program Name CS13_9.C*/
1:    #include <alloc.h>
```

```
2:   #include <stdio.h>
3:   #include <stdlib.h>
4:   int  do_b_p( long );
5:   void main(void)
6:      {
7:         int  rv;
8:         unsigned long nbr_pages = 0;
9:         unsigned long page = 0;
10:        printf("\n\nEnter number of pages to do ==> ");
11:        scanf( "%d", &nbr_pages );
12:        for( page = 1; page <= nbr_pages; page++)
13:           {
14:              rv =  do_b_p( page );
15:              if (rv ==100)
16:                 {
17:                    printf("\nAllocation error, exiting...");
18:                    exit(1);
19:                 }
20:           }
21:      printf( "\n\nDid all!\n" );
22:  }
23:  int  do_b_p( long page_nbr )
24:     {
25:        char far *book_page;     /* pointer to assign allocation to */
26:         b_page = (char *) farmalloc( 1000 );
27:         if( b_page == NULL )
28:            {
29:              printf( "\nError in line %3.3d: Could not allocate
                                                 memory.",__LINE__);
30:              return(100);
31:            }
32:          else
33:            {
34:               printf( "\n Allocation for book page %ld is ready to
                                                 use...",page_nbr);
```

```
35:            }
36:        return(0);
37:    }
```

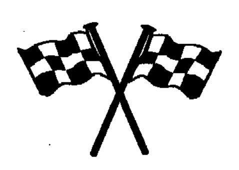
شكل ٩-١٣ استخدام الدالة faralloc

وعندتنفيذ البرنامج الموجود بالشكل (٩-١٣) يعرض النتيجة التالية :

```
Allocation for book page 9991 is ready to use...
Allocation for book page 9992 is ready to use...
Allocation for book page 9993 is ready to use...
Allocation for book page 9994 is ready to use...
Allocation for book page 9995 is ready to use...
Allocation for book page 9996 is ready to use...
Allocation for book page 9997 is ready to use...
Allocation for book page 9998 is ready to use...
Allocation for book page 9999 is ready to use...
Allocation for book page 10000 is ready to use...
Did all the pages!
```

وعن هذا البرنامج نوضح ما يلى :

فى البرنامج الموجود فى الشكل رقم ٩-١٠ فى السطر ٢٦ الدالة ()faralloc تستخدم لحجز ١٠٠٠ مكان كل مرة يتم فيها استدعاء الدالة do_b_p وهو نفس عمل الدالة ()malloc الا أن الدالة ()farmalloc تتعدى فى حجزها 64kb للتأكد من ذلك استبدل الدالة ()farmalloc فى هذا البرنامج بالدالة ()malloc ونفذ البرنامج وشاهد النتيجة تجد البرنامج باستعمال الدالة ()malloc يحجز عدد من الاماكن أقل.



# الفصل الرابع عشر
# روتينات الذاكرة
# ROM BIOS

*في هذا الفصل ستتعرف على :*

- *المسجلات ( registers) العامة للمعالج*
- *كيفية استعمال الروتينات.*
- *برنامج لمعرفة حجم الذاكرة الأساسية.*
- *استعمال ملف العناوين dos.h.*
- *برامج للتعامل مع المؤشر.*

تشتمل الذاكرة الثابتة (ROM) على مجموعة من التعليمات الضرورية لتشغيل الجهاز والتعامل مع مكوناته ، وعادةً تقوم الشركات الصانعة بوضع هذه التعليمات داخل الذاكرة الثابتة ، وتبقى هذه التعليمات ثابتة بحيث يمكن التعامل معها فى أى وقت وتسمى BIOS (Basic Input Output (System)) وتسمى كل مجموعة من التعليمات تقوم بوظيفة محددة أو روتين (routine) وهذه الروتينات هى المسؤلة عن عمليات الدخل والخرج وتتعامل لغات البرمجة مع الروتينات (أو البرامج الصغيرة) الموجودة بالذاكرة عندما نحتاج اليها دون تدخل من المبرمج، وبذلك توفر عليه وقتاً وجهداً كبيرين كما يحدث فى عمليات الدخل والخرج ولكن إستعمال هذه الروتينات مباشرة يمكنك من عمل الكثير مما لاتسطيع أن تحققه بدوال وتعليمات اللغة مباشرة. يحقق إستعمال الروتينات مزايا أخرى كثيرة وستعرفها فى هذا الفصل وقبل أن نشرح خطوات استدعاء الروتينات الموجودة فى الذاكرة ROM سنشرح المقصود بالمسجلات العامة للمعالج.

## المسجلات (registers) العامة للمعالج

يعتبر المعالج أهم جزء فى جهاز الكمبيوتر ، وتنتج هذه المعالجات أكثر من شركة ، وكل شركة تضع معالجاتها فى عائلة وتعطيها اسم فمثلا هناك شركة "موتوريلا" التى تنتج معالجات (processors) وتعطيها الاسم "موتوريلا" وبالمثل توجد شركة "انتل" وتنتج معالجات باسم انتل وأجهزة IBM تقوم على معالجات شركة انتل intel وقد بدأت شركة انتل منتجاتها ابتداء من المعالج رقم ٨٠٨٨ وكان الجهاز الذى يبنى على هذا المعالج يسمى XT ثم انتجت الشركة المعالج ٨٠٢٨٦ ثم ٨٠٣٨٦ ثم المعالج ٨٠٤٨٦ وأخيرا المعالج ٨٠٥٨٦ وعلى هذه المعالجات قامت أجهزة IBM والاجهزة المتوافقة معها

وجميع المعالجات تتفق فى التركيب العـام ، وتشـتمل المعالجـات فـى داخلهـا على أجزاء دقيقة تسمى المسجلات (registers) فما هى المسجلات ، وما هى الحاجة لمعرفة هذه المسجلات ، وما هى المسجلات العامة للمعالج ؟

*المسجلات* : هـى وحـدات ذاكـرة مـن نـوع خـاص تشـبه عناصر الذاكـرة وتسـتخدم لتخزين البيانات ، وهى التى يتم فيها أجراء العمليات الحسـابية وإرسـال المعلومـات الـى الذاكرة أو استقبال البيانات منها.

معرفة هذه المسجلات ضرورية لأن استدعاء روتينات الذاكـرة ROM يتـم مـن خلالها ، وتوضع فيها معاملات الروتينات وكذلك نتائج العمليات

## ما هى المسجلات العامة للمعالج intel ؟

ينقسم المعالج الى مجموعات من المسجلات ، من هذه المجموعات مجموعة تسمى المسجلات العامة وهى التى تهمنا هنـا مـن وجهـة نظـر البرمجـة و تـأخذ الاسـماء AX,BX,CX,DX.

وهذه المسجلات كما أشرنا هى وحدات ذاكرة من نـوع خـاص سـعة التخزيـن الطبيعية لها١٦ بت ومن مزايا أوطبيعة هذه المسجلات إمكانية التعامل معها بطريقتين :

*الطريقة الاولى* معاملة كل مسجل على أنه عنصـر واحـد سـعة تخزينـه ٢ بـايت وفى هذه الحالة تأخذ المسجلات الاسماء AX,BX,CX,DX.

*الطريقة الثانية* معاملة كل مسجل على أنه عنصرين سـعة كـل عنصـر ١ بـايت ويظهر ذلك من الجدول التالى :

| | | |
|---|---|---|
| AX | AH | AL |
| BX | BH | BL |
| CX | CH | CL |
| DX | DH | DL |

وكما ترى في الجدول السابق أن المسجل AX ينقسم الى AH يسمى (HIGH)و AL ويسمى (LOW) وهكذا BX, CX, DX وقبل أن نشرح كيفية استعمال الروتينات نذكر بعض مزاياها.

## مزايا استعمال الروتينات الموجودة بالذاكرة الثابتة

- انشاء برامج باستعمال ROM BOIS ROUTINES يعطي برنامج صغير الحجم
- يعطي برنامج سريع التنفيذ
- باستعمال ROM BOIS ROUTINES تستطيع تنفيذ كثير من العمليات التي لايمكن تنفيذها بإستعمال تعليمات اللغة مباشرة.

## كيفية استدعاء الروتينات الموجودة بالذاكرة (ROM BIOS)

تأخذ الروتينات الموجودة في الذاكرة شكل برامج كل برنامج يحتوى على مجموعة دوال تؤدى عمل متقارب كلها تدور حول خدمات معينة ، وكل برنامج يسمى interrupt وكل interrupt مخصص لموضوع معين ، فمثلا تجد interrupt مسئول عن خدمات الشاشة (video interrupt) وتختص جميع دواله بهذا الموضوع فنجد به دوال تغيير الألوان ودوال التعامل مع المؤشر وهكذا ومثال أخر هو interrupt القرص (disk interrupt) ويحتوى على دوال لجميع الوظائف المتعلقة بالاسطوانة مثل كيفية تحديد حاله القرص سليمة أم لا وكيفية الكتابة في أي مكان عليها وكثير من الخدمات التي لايمكن أن تؤدى إلا بهذه الروتينات.

وكل interrupt له رقم بالنظام السادس عشر (HEX) فمثلا video interrupt رقمه ١٠ بالنظام السادس عشر (يمكنك الرجوع إلى الجداول الخاصة بهذه الروتينات والموجودة في احد الكتب المتخصصة مثل كتاب

DOS PROGRAMMER'S REFERENCE

## الدالة ( ) int86

هى الدالة المسئولة عن استدعاء الروتينات الموجودة فى ROM والحروف INT اختصار العباره intel والرقم 86 يعنى عائلة انتل وهى ,80586 ,80486 80386, 80286, 8088 والصورة العامة للدالة هى :

```
int86(INTno,&inregs,&outregs)
```

حيث أن :

المتغير INTno هو رقم interrupt المطلوب استعماله

والمتغير &inregs هى عناوين المسجلات التى توضع فيها المدخلات

والمتغير &outregs هى عناوين المسجلات التى توضع بها المخرجات

ولتوضيح كيفية استعمال هذه الدالة فى استدعاء الروتينات نفترض روتين كالموجود بالشكل التالى :

| | |
|---|---|
| اسم الروتين | Rom Bios |
| الغرض منه | تحديد حجم الذاكرة |
| الرقم الخاص به | 12 Hex |
| المدخلات | لايوجد |
| المخرجات | توضع فى المسجل AX |

والبرنامج الموجود فى الشكل رقم (١٤-١) يستخدم الروتين السابق فى طباعة حجم الذاكرة الرئيسية.

```
0:  /*Program Name CS14_1.C */
1:  #define MEM 0x12
2:  void main (void)
3:  {
```

```
4:      struct WORDREGS
5:            {
6:                unsigned  int ax;
7:                unsigned  int bx;
8:                unsigned  int cx;
9:                unsigned  int dx;
10:               unsigned  int si;
11:               unsigned  int di;
12:               unsigned  int flags;
13:           };
14:     struct BYTEREGS
15:           {
16:               unsigned char al,ah;
17:               unsigned char bl,bh;
18:               unsigned char cl,ch;
19:               unsigned char dl,dh;
20:           };
21:     union REGS
22:           {
23:               struct WORDREGS x;
24:              struct BYTEREGS h;
25:           };
26:     union REGS regs;
27:     int size;
29:     int86(MEM,&regs,&regs);
30:     size=regs.x.ax;
31:     printf ("Memory size is %d Kilo Bytes",size);
32:  }
```

الشكل رقم ١-١٤ برنامج طباعة حجم الذاكره

ونتيجة تنفيذ هذا البرنامج كالتالى:

```
Memory size is 640 Kilo Bytes
```

وعن هذا البرنامج نوضح ما يلى :

يبدأ البرنامج الموجود فى الشكل رقم ١-١٤ فى السطر رقم ٤ بانشاء سجل بالاسم WORDREGS وعناصره هى المسجلات العامة ولكن بصورتها الأولى وهى كل مسجل سعته 2 Byte ولذلك كان نوع المتغيرات int حيث حجم الـ int هو 2 Byte ، وكذلك unsigned حتى لا يكون له اشارة وبالتالى حجم المتغير 2 Byte.

- وفى السطر رقم ١٤ أنشأنا سجل أخر باسم BYTEREGS وعناصره هى المسجلات بالصورة الثانية وهى أن كل مسجل عبارة عن قسمين فمثلا AX يتكون من AH, AL وهكذا
- وفى السطر رقم ٢١ تم تعريف UNION عناصره عبارة عن سجلان الأول هو من النوع WORDREGS والثانى من النوع BYTEREGS ونلاحظ هنا أن الـ UNION حقق الصورة المطلوبة حيث أن المسجلات فى الحالتين هما شيء واحد ولكن يمكن معاملتها بطريقتين وهذه هى طبيعة الـ UNION حيث يحجز مساحة واحدة
- وفى السطر ٢٦ أعلنا عن متغير من نوع هذا الـ UNION وأسمه regs
- وفى السطر ٢٩ تم استدعاء الدالة int86() وتم ارسال المعاملات اليها وهى رقم Interrupt وهو كلمة MEM المعرفة فى أول البرنامج بالرقم 0x12 ، ونتيجة الاستدعاء هى وضع حجم الذاكرة فى المسجل AX.
- وفى السطر رقم ٣٠ نأخذ حجم الذاكرة الموجود بـ AX ونضعه فى المتغير size ثم نطبع هذه القيمة فى السطر رقم ٣١

ويمكن تلخيص خطوات التعامل مع أى روتين من روتينات الذاكرة (ROM BIOS) فيما يلى :

١. تحديد رقم Interrupt
٢. تحديد رقم الدالة أن وجد

٣. تحديد المدخلات والمسجلات التي يوضع فيها المدخلات

٤. تحديد المخرجات والمسجلات التي يوضع فيها المخرجات

٥. استدعاء الدالة ()INT86

## استعمال ملف العناوين dos.h

في المثال السابق أنشأنا UNION بالاسم REGS يحتوى على سجلان 2 STRUCTURES) يمثلان المسجلات العامة للمعالج بالصورتين. ويتم إعلان هذا الـ Union في كل برنامج يستعمل روتينات الذاكرة ، ونظرا لأن هذا الجزء ثابت فقد تم انشاء هذا UNION في الملف DOS.H وليس عليك الا أن تكتب السطر#include<dos.h> في أول البرنامج وبالتالي يوفر عليك انشاء هذا الـ Union ويصبح البرنامج الموجود في الشكل رقم (١-١٤) بسيط كما في الشكل رقم (٢-١٤) هو ويعطي نفس النتيجة

```
0:   /*Program Name CS11_2.C*/
1:   #include <dos.h>
2:   #define MEM 0x12
3:   void main (void)
4:     {
6:          union REGS regs;
7:          int size;
8:          clrscr();
9:          int86(MEM,&regs,&regs);
10:         size=regs.x.ax;
11:         printf ("Memory size is %d Kbytes",size);
12:         getch();
13:  }
```

الشكل رقم ٢-١٤ استعمال الملف dos.h

## تغيير حجم المؤشر Seting the cursor size

الروتين التالى يقوم بتغيير حجم المؤشر حيث ينقسم المؤشر(cursor) الى ١٣ خط إفقى يمكن لنا تحديد خط بداية المؤشر وخط النهاية وبالتالى تحديد حجم المؤشر ويتم تحديد الحجم بوضع رقم خط البداية فى المسجل CH ورقم خط النهاية فى المسجل CL ثم إستدعاء الروتين فيتغيير حجم المؤشر.

| | |
|---|---|
| إسم الروتين | Cursor Size |
| الغرض منه | تغيير حجم المؤشر |
| رقم الروتين | 0x10 hex |
| المدخلات | المسجل CH يأخذ بداية المؤشر ، المسجل CL يأخذ نهاية المؤشر |
| المخرجات | لايوجد (تغير شكل المؤشر) |

رقم الدالة 01 ويوضع فى المسجل Ah

والبرنامج الموجود بالشكل رقم ٣-١٤ يقوم باستعمال الروتين السابق فى تغير حجم المؤشر.

```
0:   /*Program Name CS14_3.C */
1:   #include <dos.h>
2:   #define VIDEO 0x10
3:   #define CURSIZE 1
4:   void main (int argc,char *argv[])
6:  union REGS regs;
7:  int start,end;
8:   if (argc!=3)
9:       {
```

```
10:         printf ("\n Example usage: C>setcurs 12 13");
11:         exit();
12:      }
13:    start=atoi(argv[1]);
14:    end=atoi(argv[2]);
15:    regs.h.ch=(char)start;
16:    regs.h.cl=(char)end;
17:    regs.h.ah=CURSIZE;
18:    int86(VIDEO,&regs,&regs);
19: }
```

الشكل رقم ٣-١٤ تغير حجم المؤشر

وعن هذا البرنامج نوضح ما يلى :

قم بترجمة البرنامج وتنفيذه باستخدام الأمر التالى من محث DOS :

CS14_3  2 6

حيث أن المتغير argc يحتوى على ٣ معادلات والمعاملات تخزن فى المصفوفة []argv بالترتيب التالى :

argv[0],argv[1],argv[2]

وعلى ذلك فإن القيمة ٢ التى استقبلها البرنامج عند استدعائه تخزن فى العنصر argv[1] والقيمة ٦ تخزن فى العنصر argv[2].

- وفى السطر رقم ١٣ الدالة () atoi تحول المعامل الاول (القيمة ٢) الى رقم صحيح لانه يستقبل على أنه STRING ويخزنه فى المتغير start وبالمثل المتغير end.
- فى السطر رقم ١٧ ، نضع فى المسجل ah قيمة المتغير CURSIZE المعرفة فى أول البرنامج بالقيمة ١ وهو رقم الدالة الخاصة بتغير حجم المؤشر.

- وفى السطر رقم ١٨ يتم استدعاء الدالـة ()int86 بالمعـاملات التـى تـم ضبطهـا فتكون النتيجة هى تغير حجم المؤشر بالقيم المحددة كمعاملات للبرنامج.
وتلاحظ فى هذا الأمر أن استـدعاء البرنـامج للتنفيـذ يتـم بمعـاملين همـا ٢ ، ٦ وبالرجوع إلى البرنامج الموجود فى شكل ٣-١٤ تجد أن الدالة main تحتوى على متغيرين هما Argv , argc

## اخفاء المؤشر

من الاستخدمات المتاحة للروتين السـابق الـذى اسـتعمل لتغيـير حجـم المؤشـر اخفاء المؤشر وذلك بوضع القيمة 0x20 فى المسجل ch واستدعاء نفس الروتين.

والبرنامج الموجود بالشكل رقم (٤-١٤) . ويقوم باخفاء المؤشر

أكتب هذا البرنامج ونفذه وشاهد النتيجة :

```
0:   /*Program Name CS14_4.C*/
1:   #include <dos.h>
2:   #define VIDEO 0x10
3:   #define CURSIZE 1
4:   #define STOPBIT 0x20
5:   void main (int argc,char *argv[])
6:     {
7:       union REGS regs;
8:       regs.h.ch=STOPBIT;
9:       regs.h.ah=CURSIZE;
10:      int86(VIDEO,&regs,&regs);
11:    }
```

الشكل رقم ٤-١٤ اخفاء المؤشر

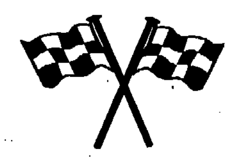
## دوال لغة C التى تستعمل ROM BOIS مباشرة

بالاضافة لامكانية استعمال الروتينات الموجودة فى ROM توجد مجموعة من الدوال قامت على هذه الروتينات بحيث توفر علينا كتابة هذه الروتينات من البداية منها الدوال التالية :

| الدالة | الغرض منها |
|---|---|
| _bios_disk | تقوم بخدمات كلا من القرص الصلب (HARD DISK) والمرن |
| _bios_equiplist | تقوم بجميع الاختبارات المطلوبة على مكونات الجهاز. |
| _bios_keybrd | تحقق جميع خدمات لوحة المفاتيح |
| _bios_memsize | تعطينا معلومات عن الذاكرة المتاحة |
| _bios_printer | تحقق جميع خدمات الة الطباعة |
| _bios_serialcom | تحقق خدمات منفذ التوالى الخاص بالجهاز (SERIAL PORT). |
| _bios_timeofday | التعامل مع ساعة الجهاز |

ولا يتسع المجال هنا لتفصيلها ولكن يمكن لك أن ترجع الى مرجع الدوال الخاص بلغة C (LIB REFERENCE) لمعرفتها واستعمالها وقت الحاجة

*أمثلة* : هناك مجموعة من البرامج النص الكامل لها موجود على الاسطوانة المصاحبة للكتاب لك أن تفتح هذه البرامج وتنفذها مباشرة.






يحتوى هـذا الفصـل علـى برنـامج متكـامل يشـتمل علـى معظم الموضوعات التى ناقشناها فى الفصول السابقة .

وهذا البرنـامج عبـارة عـن برنـامج قواعـد بيانـات بسـيط يحتوى على معظم العمليات المطلوبة حيث يمكننـا البرنـامج مـن إضافة أو حـذف بيانـات أو التعديـل فـى البيانـات وكذلـك إظهـار البيانات على الشاشة او طباعتها على الة الطباعة.

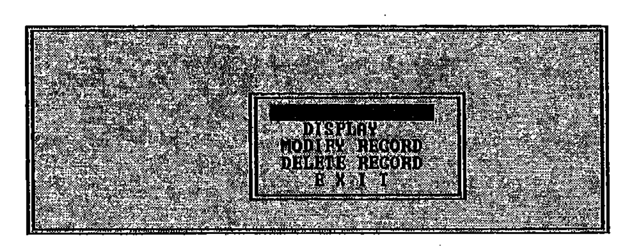
ويشتمل هذا الفصل على:

- متابعة مراحل تنفيذ ابرنامج
- عرض نص البرنامج
- ملخص سريع عن أجزاء البرنامج

## أولا: متابعة تنفيذ البرنامج

اكتب البرنامج كما هو مكتوب فى نص البرنامج أو إفتح ملف البرنامج من القرص المصاحب للكتاب وتابع معنا مراحل التنفيذ.

عند تنفيذ البرنامج تظهر اول شاشة وهى الشاشة الرئيسية للبرنامج كما فى الشكل ١-١٥ وتحتوى على إختيارات البرنامج الرئيسية وبإستخدام مفاتيح الاسم نحرك الشريط المضاء ونحدد الاختيار المطلوب تنفيذه ولنبدأ بالاختيار الاول ثم الضغط على مفتاح الادخال نحصل على شاشة ادخال البيانات كما فى الشكل ٢-١٥




شكل رقم ١-١٥

```
Enter name : azab mohamed azab
Enter number: 111
Enter TEL: 2219047

      Add another record (y/n)
```

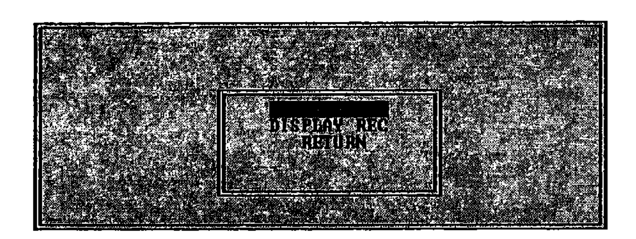
شكل رقم ٢-١٥

ادخل بيانات كما فى الشكل وفى النهاية تحصل على رسالة

**Add another record (y/n)**

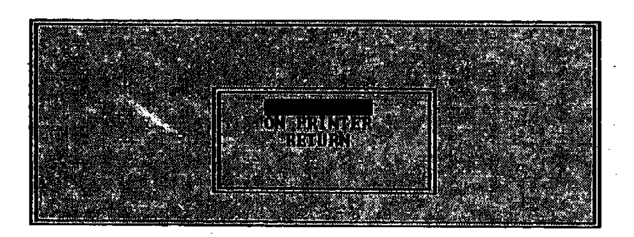
أدخل الحرف n وبالتالى تعود إلى الشاشة الرئيسية

تحرك بالاسهم إلى الاختيار الثانى وهو display ثم إضغط مفتاح الادخال تحصل على الشكل ٣-١٥ وبه إختيابن هما display record و display all




شكل ٣-١٥

حرك الشريط المضاء إلى الاختيار الأول ثم إضغط على مفتاح الادخال تحصل على شاشه بها اختيارين هما on printer و on screen كما بالشكل ٤-١٥ اختر الاول ثم اضغط على مفتاح الادخال تحصل على شاشة التقارير على الشاشة كما بالشكل ٥-١٥




شكل ٤-١٥

بعد متابعة عرض البيانات على الشاشة إضغط اى مفتاح ترجع الى شاشة العرض (display) مرة اخرى ، اختر الاختيار الثانى وهو display rec ثم إضغط مفتاح الادخال تحصل على شاشة عرض بيانات سجل كما فى الشكل ٦- ١٥ اكتب الاسم المراد عرض بياناته ثم إضغط مفتاح الادخال تظهر بيانات السجل المطلوب كما

فى الشكل ٧-١٥

```
  NAME          NUMBER1          NUMBER2
.... azab mohamed a.....111.......2219047......
```

شكل ٥-١٥




شكل ٦-١٥

```
******************************

        Name:azab mohamed azab
        Number:111
        TEL:2219047
***again(y/n)>> ********************
```

شكل ٧-١٥

وبنفس الاسلوب يمكن لك متابعة تنفيذالبرنامج بتفيذ الاختبارات الباقية ومشاهدة النتائج.

## نص البرنامج

فيما يلى النص الكامل للبرنامج

```
#define TRUE 1
#define NUM 5
#define NUM2 3
#define CLEAR   "\x1B[2J"
#define ERASE   "\x1B[K"
```

```
#define NORMAL   "\x1B[0m"
#define REVERSE  "\x1B[7m"
#define HOME    "\x1B[10;5f"
#define BOTTOM     "\x1B[20;1f"
#define U_ARRO    72
#define D_ARRO     80
#define INSERT     82
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <dos.h>
/****************functions declerations  part (1) *****************.*******/
void dispp();
void displayallp(void);
void action2(int pos2);
void action3(int pos2);
void ex();
void add(void);
void action4 (int pos);
void displayall(void);
void displayrec(void);
void modify(void);
void del(void);
void disp();
void call();
void action(int);
void display (char *arr[],int size,int pos);
char getcode(void);

/*****************************************************************************/
 char ag;
```

```
char namec[40];
char ch,y1;
/****************************** THE RECORD STRUCTRURE ***********/
        struct
          {
          char name[40];
          char  agnumb[10];
          char  tel[20];
          } agent;
/***************************************************************************/
 char numstr[81];
 FILE *fptr,*fptr2;
/***************************** BOX FUCTIONS ******************************/
 void set_window1()
 {
 window (2,2,79,9);
 }
 void set_window2()
 {
 window(2,12,79,23);
 }
 void fancy_box(int x1,int y1,int x2,int y2)
        {
        int i;
        gotoxy(x1,y1);putch(201);
        for(i=x1+1;i<x2;i++) putch(205);
        putch(187);
        for(i=y1+1;i<y2;i++) {
        gotoxy(x1,i);putch(186);
        gotoxy(x2,i);putch(186);
        }
        gotoxy(x1,y2);putch(200);
        for (i=x1+1;i<x2;i++) putch(205);
        putch(188);
```

```
}
void box_it(int x1,int y1,int x2,int y2)
{
int i;
gotoxy(x1,y1);putch(218);
for (i=x1+1;i<x2;i++) putch (196);
putch(191);
for (i=y1+1;i<y2;i++) {
gotoxy(x1,i);putch(179);
gotoxy(x2,i);putch(179);
}
gotoxy(x1,y2);putch(192);
for (i=x1+1;i<x2;i++) putch(196);
putch(217);
}
/*****************************************************************************/
/**************************** MENUS   STRUCTURES ********************/
static char *item[NUM2]=
                    { "DISPLAY ALL",
                      "DISPLAY REC",
                      "  RETURN  " };
static char *itemp[NUM2]=
                    { "ON SCREEN ",
                      "ON PRINTER",
                      "  RETURN  " };
static char *iteme[NUM2]=
                    { "E  N  D",
                      "DOS SHELL",
                      "  RETURN " };
             static char *items[NUM]=
                { " ADD RECORD   ",
                  "  DISPLAY    ",
                  " MODIFY RECORD ",
                  " DELETE RECORD ",
```

```
                    "  EXIT  "};

/****************************************************************************/
/********************* MAIN PROGRAM  main() function **************/
  void main (void)
          {
/******************** COLORS  AND BOXES***************************/
      textbackground(BLUE);
      textcolor(WHITE);
      clrscr();
      fancy_box(1,1,80,24);
      fancy_box(8,8,60,20);
      fancy_box(28,12,47,18);
          clrscr();
          call();
      }
      void call()
      {
      int curpos,s;
      curpos=0;

        while (TRUE)
          {
          textbackground(BLUE);
      textcolor(WHITE);
      clrscr();

      clrscr();
      fancy_box(1,1,80,24);
      fancy_box(8,8,60,20);
      fancy_box(28,12,47,18);
/****************************************************************************/
/*************** DISPLAY MAIN MENU  BY USING display()*************/
          display(items,NUM,curpos);
```

```
switch (getcode())
        {
        case U_ARRO:
          if(curpos>0)
          {
           -- curpos;
          sound(800);
          delay(20);
nosound();
          }
          else
          curpos+=(NUM-1);
          break;
        case D_ARRO:
          if(curpos<NUM-1)
          {
          ++curpos;
          sound(900);
          delay(20);
           nosound();
            }
          else
          curpos -=(NUM-1);
          break;
        case '\r':
           for (s=0;s<10;s++)
             {
           sound(s*100);
           delay(50);
           nosound();
             }
          action(curpos);
           break;
          }
```

```
        }
      }
/**************************** END OF main() FUNCTION ****************/

/******************************** display  function()***********************/
   void display (char *arr[],int size,int pos)
    {
    int j;
        printf (HOME);
        for (j=0;j<size;j++)
          {
          if(j==pos)
              {
              textcolor(2);
              textbackground(4);
              }
              gotoxy(30,13+j);
              cprintf ("%s\n",*(arr+j));
              textcolor(WHITE);
              textbackground(BLUE);
              }
              printf (BOTTOM);
              }
/*****************************************************************************/
/*************************** getcode(void) function ************************/
        char getcode(void)
          {
          int key;
          if ((key=getch() )==0)
              return (getch() );
          else if (key=='\r')
              return(key);
          else
          return (0);
```

```
            }
/**************************** action(int pos) function ***********************/
      void action(int pos)
        {
         printf (ERASE);
             switch (pos)
               {
               case 0:
                   add();
                   break;
               case 1:
                   disp();
                   break;
               case 2:
                   modify();
                   break;
               case 3:
                   del();
                   break;
               case 4:
                   ex();
                   break;
                   default:
                          printf("\n unkowen");
                   }

               }
/****************************** disp() function ******************************/
            void disp()
        {
        int curpos2=0;
        clrscr();

      while (TRUE)
```

```
        {
    textbackground(BLUE);
    textcolor(WHITE);
    clrscr();
 fancy_box(1,1,80,24);
    fancy_box(8,8,60,20);
    fancy_box(25,12,45,18);
            display(item,NUM2,curpos2);
              switch (getcode())
                  {
                  case U_ARRO:
                   if(curpos2>0) -- curpos2;
                   else
                   curpos2+=(NUM2-1);
                   break;
                  case D_ARRO:
                   if(curpos2<NUM2-1) ++curpos2;
                   else
                   curpos2 -=(NUM2-1);
                   break;
                  case '\r':
                   action2(curpos2);
                   break;
                   }
                  }
                  }
/************************** action2(int pos) ***********************************/
      void action2(int pos)
       {
        int s;
        printf (ERASE);
            switch (pos)
              {
              case 0:
```

```
                    dispp();
                    break;
                case 1:
                    displayrec();
                    break;
                case 2:
                    for (s=0;s<10;s++)
                      {
                      sound(s*100);
                      delay(50);
                      nosound();
                      }
                    call();
                    break;

               }
           }
/*************************** dispp()  function  ******************************/
           void dispp()
      {
      int curpos2=0;
      clrscr();
    while (TRUE)
           {
           textbackground(BLUE);
    textcolor(WHITE);
    clrscr();
      fancy_box(1,1,80,24);
    fancy_box(8,8,60,20);
    fancy_box(25,12,45,18);
           display(itemp,NUM2,curpos2);
              switch (getcode())
                  {
                  case U_ARRO:
```

```
                if(curpos2>0) -- curpos2;
                else
                curpos2+=(NUM2-1);
                break;
               case D_ARRO:
                if(curpos2<NUM2-1) ++curpos2;
                else
                curpos2 -=(NUM2-1);
                break;
               case '\r':
                action4(curpos2);
                break;
                }
                }
                }
/**************************** action4 (int pos) function *******************/
     void action4 (int pos)
      {
       int s;
       printf (ERASE);
            switch (pos)
              {
              case 0:
                  displayall();
                  break;
              case 1:
                  displayallp();
                  break;
              case 2:
                  for (s=0;s<10;s++)
                     {
                     sound(s*100);
                     delay(50);
                     nosound();
```

```
                    }
                  call();
                  break;

               }
            }

/************************ ex() function ******************************************/
            void ex()
          {
          int curpos2=0;
          clrscr();

        while (TRUE)
            {
            textbackground(BLUE);
        textcolor(WHITE);
        clrscr();
          fancy_box(1,1,80,24);
        fancy_box(8,8,60,20);
        fancy_box(25,12,45,18);
             display(iteme,NUM2,curpos2);
               switch (getcode())
                   {
                   case U_ARRO:
                    if(curpos2>0) -- curpos2;
                    else
                    curpos2+=(NUM2-1);
                    break;
                   case D_ARRO:
                    if(curpos2<NUM2-1) ++curpos2;
                    else
                    curpos2 -=(NUM2-1);
                    break;
```

```
                case '\r':
                 action3(curpos2);
                 break;
                 }
                }
                }
/******************************** action3(int pos) function ****************/
        void action3(int pos)
         {
         int s;
         printf (ERASE);
              switch (pos)
                {
                clrscr();
                case 0:
                     clrscr();
                     cputs("\n\n\n\n\n\n\n\t");
                     cputs("..... thank's and goodby ENG.AZAB....");
                     for (s=5;s<10;s++)
                         {
                       cputs("\n\n\n\n\n\n\n\t");
                     cputs("..... thank's and goodby ENG.AZAB....");
                       sound(s*50);
                       delay(450);
                       nosound();
                        sound(s*150);
                       delay(350);
                       nosound();

                       sound(s*120);
                       delay(450);
                       nosound();

                       sound(s*120);
```

```
                    delay(450);
                    nosound();
                    sound(s*120);
                    delay(450);
                    nosound();

                     }
                exit(0);
                break;
            case 1:
                clrscr();
                system("command.com");
                printf ("\n\n\n TYPE EXIT TO RETUTN TO MAIN
                                                        PROG..");
                break;
            case 2:
                for (s=0;s<10;s++)
                   {
                  sound(s*100);
                  delay(50);
                  nosound();
                   }
                call();
                break;

          }
       }
/*********************** add(void) function ********************************/
     void add(void)
         {
    int s;
   clrscr();

    if( (fptr=fopen("agents.rec","ab"))==NULL)
```

```
    open file agents.rec ");exit(
        do
        {
        clrscr();
        cputs("\r\n\n");
        cputs ("  Enter name : ");
        gets(agent.name);
        cputs ("\r\n  Enter number: ");
        gets(agent.agnumb);
/*      agent.agnumb=atoi(numstr); */
        cputs ("\r\n  Enter  tel: ");
        gets(agent.height);
/*      agent.height=atof(numstr); */
        fwrite(&agent,sizeof(agent),1,fptr);
        cputs ("\r\n\n      Add an other record (y/n)");
        }
        while(toupper(getche())=='Y');
        fclose(fptr);
        for (s=0;s<10;s++)
                {
                sound(s*100);
                delay(50);
                nosound();
                }
        clrscr();
    }

/************************** displayrec(void) function **********************/
    void displayrec(void)
        {
          int s;
                clrscr();
    if( (fptr=fopen("agents.rec","rb"))==NULL)
      { printf ("can't open file agents.rec ");exit(1);}
```

```
    clrscr();
     cprintf ("\n  Enter name:");
     gets(namec);
    while ((fread(&agent,sizeof(agent),1,fptr)==1))
    {
      if (strcmp(namec,agent.name)==0)
      {
  textattr(80);
  clrscr();
     cputs ("\r\n      ********************************");
     cputs("\r\n\n\n");
     cputs ("          Name:");
     cputs(agent.name);
     cputs ("\r\n\n");
     cputs("          Number:");
     cprintf ("%s",agent.agnumb);
     cputs ("\r\n\n");
     cputs("         tel:");
     cprintf ("%s",agent.height);
     cputs ("\r\n      ***again(y/n)>> **********************");
     }
    }
           fclose(fptr);
           y1=getch();
          for (s=0;s<10;s++)
                  {
                  sound(s*100);
                  delay(50);
                  nosound();
                  }
           clrscr();
     }
/*********************** displayallp(void) function ***********************/
     void displayallp(void)
```

```
{
    int s;
    int n=5;
    clrscr();
  if( (fptr=fopen("agents.rec","rb"))==NULL)
   { printf ("can't open file agents.rec ");exit(1);}
fprintf (stdprn," ____________________________________________\n");
fprintf (stdprn,"\t\t NAME      NUMBER1      NUMBER2 \n");
fprintf (stdprn,"____________________________________________\n");
   while ((fread(&agent,sizeof(agent),1,fptr)==1))
        {
   fprintf (stdprn,"..... %s.....",agent.name);
   fprintf (stdprn,".....%s......",agent.agnumb);
   fprintf (stdprn,".....%s......\n",agent.height);
   fprintf (stdprn,"\t ________________________________\n");
   sound(200*n);
   delay(200);
   n++;
    if (n==14)
    {
    nosound();
   getch();
   clrscr();
   n=5;
fprintf (stdprn," ____________________________________________\n");
fprintf (stdprn,"\t\t NAME        NUMBER1      NUMBER2 \n");
fprintf (stdprn," ____________________________________________\n");
   }
  }
          fclose(fptr);
          getch();
       nosound();
       for (s=0;s<10;s++)
           {
```

```
                    sound(s*100);
                    delay(50);
                    nosound();
                     }
               clrscr();
               }
/******************** displayall(void) *****************************************/
               void displayall(void)
          {
             int s;
             int n=5;
             clrscr();
        if( (fptr=fopen("agents.rec","rb"))==NULL)
          { printf ("can't open file agents.rec ");exit(1);}
          textattr(30);
   clrscr();
 printf ("________________________________________________________\n");
 printf ("\t\t NAME      NUMBER1       NUMBER2 \n");
 printf ("________________________________________________________\n");
          while ((fread(&agent,sizeof(agent),1,fptr)==1))
               {
         gotoxy(5,n);
         cprintf (".....  %s.....",agent.name);
        gotoxy(25,n);
         cprintf (".....%s......",agent.agnumb);
         gotoxy(35,n);
         cprintf (".....%s......\n",agent.height);
         printf ("\t______________________________________________\n");
          sound(200*n);
          delay(200);
          n++;
           if (n==14)
           {
           nosound();
```

```
        getch();
        clrscr();
        n=5;
printf ("______________________________________________\n");
printf ("\t\t NAME        NUMBER1        NUMBER2 \n");
printf ("   ______________________________________________\n");
        }
        }
                fclose(fptr);
                getch();
            nosound();
            for (s=0;s<10;s++)
                {
                sound(s*100);
                delay(50);
                nosound();
                }
            clrscr();
            }
/****************************** modify(void) function ****************/
        void modify(void)
            {
            int s;
            clrscr();
        if( (fptr=fopen("agents.rec","rb"))==NULL)
         { printf ("can't open file agents.rec ");exit(1);}
         if( (fptr2=fopen("agents2.rec","wb"))==NULL)
         { printf ("can't open file agents2.rec ");exit(1);}
        clrscr();
        cprintf ("\n Enter name:");
        gets(namec);
        while ((fread(&agent,sizeof(agent),1,fptr)==1))
        {
         if (strcmp(namec,agent.name)==0)
```

```
        {
    textattr(80);
    clrscr();
      cputs ("\r\n      ************ OLD DATA ******************");
      cputs("\r\n\n\n");
      cputs ("          Name:");
      cputs(agent.name);
      cputs ("\r\n\n");
      cputs("          Number:");
      cprintf ("%s",agent.agnumb);
      cputs ("\r\n\n");
      cputs("          tel:");
      cprintf ("%s",agent.height);
      cputs ("\r\n      ************** PRESS ANY KEY ************");
           getch();
           clrscr();
           cputs("\r\n=========== N E W  D A T A=======\r\n");
           cputs (" Enter name2 : ");
           gets(agent.name);
           cputs ("\r\n Enter number2: ");
           gets(agent.agnumb);
/*         agent.agnumb=atoi(numstr);*/
           cputs ("\r\n Entertel2: ");
           gets(agent.height);
/*         agent.height=atof(numstr);  */
        }
           fwrite(&agent,sizeof(agent),1,fptr2);
           }
                fclose(fptr);
                fclose(fptr2);
            system("del agents.rec");
            system("ren agents2.rec agents.rec");
                getch();
           for (s=0;s<10;s++)
```

```
                        {
                        sound(s*100);
                        delay(50);
                        nosound();
                        }
        clrscr();
        }
/**************************** del(void) function *************************/
              void del(void)
              {
              int s;
              textattr(80);
              clrscr();
      if( (fptr=fopen("agents.rec","rb"))==NULL)
        { printf ("can't open file agents.rec ");exit(1);}
        if( (fptr2=fopen("agents2.rec","wb"))==NULL)
        { printf ("can't open file agents2.rec ");exit(1);}
      clrscr();
       printf ("\n   Enter name:");
       gets(namec);
     do
     {
      while ((fread(&agent,sizeof(agent),1,fptr)==1))
      {
        if (strcmp(namec,agent.name)==0)
        {
      textattr(80);
     clrscr();
               cputs("\r\n\n\n");
       cputs ("        Name:");
       cputs(agent.name);
       cputs ("\r\n\n");
       cputs("        Number:");
       cprintf ("%s",agent.agnumb);
```

```
        cputs ("\r\n\n");
        cputs("        tel:");
        cprintf ("%s",agent.height);
        cputs ("\r\n      ************ PRESS ANY KEY ************");
             cputs ("\n ARE YOU SURE (Y/N)======>");
              ch=getche();
              }
             }
             fwrite(&agent,sizeof(agent),1,fptr2);
             }
             while (toupper(ch)!='Y');
                     fclose(fptr);
                     fclose(fptr2);
              system("del agents.rec");
              system("ren agents2.rec agents.rec");
                     getch();
                     for (s=0;s<10;s++)
                        {
                        sound(s*100);
                        delay(50);
                        nosound();
                        }
        clrscr();
        }
/*****************************************************************************/
```

شكل ٨-١٥ نص البرنامج

## الشرح

لقد قسمنا البرنامج إلى الأجزاء التالية

******************** functions declerations- part 1**********************

وهوخاص بالاعلان عن الدوال التى سوف تستخدم فى البرنامج.

****************THE RECORD STRUCTRURE- part2*********************

وهو تركيب سجل البيانات الذى يستعمل فى تسجيل البيانات فى ملف قواعد البيانات

**********************BOX FUCTIONS- part3*****************************

ويحتوى على مجموعة دوال تقوم برسم مستطيل (اطار) يتم استدعائها مع الشاشات

*********************MENUS STRUCTURES-part4**********************

تركيب السجلات التى تحتوىعلى محتويات شاشات البرنامج مثل الشاشة الرئيسية وشاشة الاظهار .....

****************** MAIN PROGRAM main() function-part5***************

بداية الدالة الرئيسية وبها الاجزاء التالية

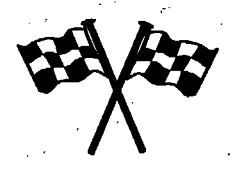
- COLORS AND BOXES و فيها نحدد الالوان ونرسم اطار الشاشة الرئيسية
- DISPLAY MAIN MENU BY USING display() وفيها استدعاء دالة الاظهار display() كما فى الفصل السابع والتى تقوم باظهار الشاشة الرئيسية
- سطور دالة displayوهىعبارة عن انشاء دالة اظهار القائمة الرئيسية كما فى الفصل السابع
- الدالة getcode(void) وتقوم باستقبال اختيار المستخدم وارساله الىالدالة الرئيسية التى تقوم بدورها بتحديد هذا الاختيار وتستدعى الدالة المنسابة لهذا الاختيار
- الدالة action(int pos) التى تستدعيها الدالة الرئيسية لتنفيذ اختيار المستخدم حيث تقوم هذه الدالة باستعمال رقم الاختيار الذى تحدده الدالة الرئيسية باستدعاء الدالة المقابلة للاختيار
- الدالة disp() و تقوم هذه الدالة باظهار شاشة الاظهار رقم (١)
- لدالة action2(int pos) و تقوم باستدعاء الدالة المناسبة لاختيارات شاشة

الاظهار

- الدالة dispp() هىدالة اظهار البيانات على الة الطباعة أم على الشاشة
- الدالة add(void) وهى دالة اضافة بيانات
- الدالة displayrec(void) هى دالة اظهار بيانات سجل
- الدالة displayallp(void)وهى دالة اظهار جميع البيانات على الة الطباعة
- الدالة displayall(void) وهى دالة اظهار جميع البيانات على الشاشة
- modify(void) function دالة التعديل فى بيانات سجل
- الدالة del(void) هى دالة حذف بيانات سجل

بهذا التطبيق نكون قد انتهينا معك من فصول كتاب المرجع الاساسى للغة C والذى نرجو أن يؤدى الثمرة المرجوة منه وأن يعود على قارئية بالنفع ،على أن نلتقى قريبا مع برمجة النوافذ و VISUAL C++

*(وأخر دعوانا أن الحمد لله رب العالمين)*



# الملاحق

*الملحق الأول  التعامل مع البيئة المتكاملة لكتابة برنامج C*

*الملحق الثاني  الكلمات المحجوزة فى لغة  C*

*الملحق الثالث شفرة تبادل المعلومات ASCII*

# الملحق الأول

# التعامل مع محرر كتابة البرامج

## يتناول هذا الملحق الموضوعات التالية:

- **كيفية تشغيل بيئة C والخروج منها**
- **الأوامر المستخدمة أثناء تحرير البرنامج**

## تشغيل بيئة C والخروج منها

للدخول الى حزمة برنامج C أكتب الأمر الآتى:

```
c:\>tc
```

وذلك بفرض أن حزمة البرامج الخاصة بمترجم لغة C موجودة تحت الدليل tc
من محث DOS اكتب الأمر DIR ، ستظهر لك الأدلة التالية:

```
bin
include
lib
bgi
examples
```

- الفهرس bin يحتوى على الملفات التنفيذية للغة مثل ملف التشغيل tc.exe
- الفهرس include يحتوى على جميع الملفات التى لها الامتداد h والتى نسميها header files والتى يوجد بها تعريف جميع دوال لغة C
- الدليل lib يحتوى على ملفات مكتبة دوال لغة C والتى يتم ربطها مع أى برنامج حسب الدوال المستخدمة

- الدليل bgi يحتوى على الملفات الخاصة بتهيئة الجهاز للرسم وكذلك ملفات الفونتات المعرفة باللغة
- الدليل examples يحتوى على مجموعة من الامثلة المتنوعة التى يمكن فتحها وتنفيذها

ولكى تتمكن من تشغيل بيئة كتابة البرنامج بسهولة ، وكذلك ترجمة وتنفيذ البرنامج بيسر أضف هذا السطر في الملف autoexec.bat

```
path =c:\tc\bin;c:\tc\lib;c:\tc\include;
```

للخروج من البيئة المتكاملة لكتابة البرنامج اما أن تضغط مفتاح F10 ثم تختر أمر Quit أو تضغط المفتاحين Alt+x معا.

ولمشاهدة نتيجة البرنامج اضغط مفتاحى ALT+F5

## الأوامر المستخدمة أثناء تحرير البرنامج

### *أولاً أوامر تحريك المؤشر*

| الأمر أو المفتاح | الوظيفة |
|---|---|
| Ctrl-S or Left arrow | التحرك لليسار حرف واحد |
| Ctrl-D or Right arrow | التحرك لليمين حرف واحد |
| Ctrl-A | التحرك لليسار كلمة واحدة |
| Ctrl-F | التحرك لليمين كلمة واحدة |
| Ctrl-E or Up arrow | التحرك سطر لأعلى |
| Ctrl-X or Down arrow | التحرك سطر لأسفل |
| Ctrl-W | طى الشاشة لأعلى |
| Ctrl-Z | طى الشاشة لأسفل |

| الأمر أو المفتاح | الوظيفة |
|---|---|
| Ctrl-R or PgUp | صفحة لأعلى |
| Ctrl-C or PgDn | صفحة لأسفل |

### ثانيا: أوامر اضافة وحذف حرف أو كلمة أو سطر

| الأمر أو المفتاح | الوظيفة |
|---|---|
| Ctrl-V or Ins | الكتابة في وضع الأدراج |
| Ctrl-N | ادراج سطر |
| Ctrl-Y | حذف سطر |
| Ctrl-QY | حذف إلى نهاية السطر |
| Ctrl-H or Backspace | حذف حرف لليسار |
| Ctrl-G or Del | حذف حرف |
| Ctrl-T | حذف كلمة لليمين |

### ثالثا: أوامر التعامل مع عدة سطور

| الأمر أو المفتاح | الوظيفة |
|---|---|
| Ctrl-K B | بداية تعليم مجموعة سطور |
| Ctrl-K K | نهاية تعليم مجموعة سطور |
| Ctrl-K T | تعلم كلمة واحدة |
| Ctrl-K C | نسخ مجموعة سطور تم تعليمها |
| Ctrl-K V | نقل مجموعة سطور تم تعليمها |
| Ctrl-K Y | حذف مجموعة سطور تم تعليمها |
| Ctrl-K R | قراءة ملف أو مجموعة سطور من القرص |

| الأمر أو المفتاح | الوظيفة |
|---|---|
| Ctrl-K W | كتابة مجموعة سطور الى القرص |
| Ctrl-K H | ازالة تعليم مجموعة سطور |
| Ctrl-K P | طباعة مجموعة سطور معلمة |
| Ctrl-K I | ضبط هامش مجموعة سطور معلمة |
| Ctrl-K U | الغاء ضبط هامش مجموعة سطور معلمة |

*رابعاً أوامر البحث والاستبدال*

| الأمر أو المفتاح | الوظيفة |
|---|---|
| Ctrl-Q F | بحث |
| Ctrl-Q A | بحث واستبدال. |
| Ctrl-L | تكرار آخر بحث |
| Esc | الغاء العملية |

*خامسا: أوامر متنوعة*

| الأمر أو المفتاح | الوظيفة |
|---|---|
| F10 | فتح سطر القوائم |
| Ctrl-K S or F2 | حفظ |
| F3 | ملف جديد |
| Alt-F3 | لاغلاق النافذة النشطة |
| Ctrl-I or Tab | جدولة |
| Ctrl-O I | محاذاة تلقائية |
| Ctrl-Q L | استرجاع سطر |

# الملحق الثانى

# الكلمات المحجوزة فى لغة C

**C Language Keywords**

| | | | |
|---|---|---|---|
| asm | return | auto | float |
| for | while | break | short |
| case | extern | goto | signed |
| sizeof | char | if | static |
| const | struct | continue | int |
| switch | default | typedef | do |
| union | double | long | unsigned |
| else | void | enum | volatile |
| register | | | |

# الملحق الثالث

## شفرة تبادل المعلومات ASCII

| ASCII Value<br>الشفرة | Character<br>الحرف |
|---|---|
| 000 | (null) |
| 001 | ☺ |
| 002 | ● |
| 003 | ♥ |
| 004 | ♦ |
| 005 | ♣ |
| 006 | ♠ |
| 007 | (beep) |
| 008 | ◘ |
| 009 | (tab) |
| 010 | (line feed) |
| 011 | (home) |
| 012 | (form feed) |
| 013 | (carriage return |
| 014 | ♫ |
| 015 | ☼ |
| 016 | ► |
| 017 | ◄ |
| 018 | ↕ |
| 019 | !! |
| 020 | π |
| 021 | § |
| 022 | ▬ |
| 023 | ↨ |
| 024 | ↑ |
| 025 | ↓ |
| 026 | → |
| 027 | ← |
| 028 | (cursor right) |
| 029 | (cursor left) |
| 030 | (cursor up) |
| 031 | (cursor down) |

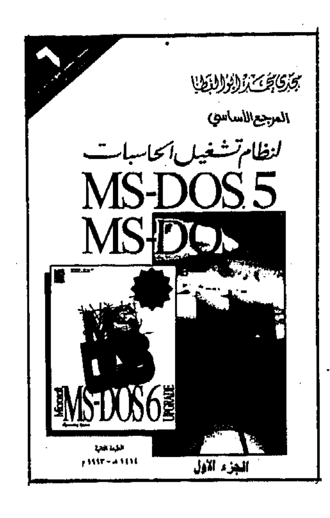
| ASCII Value الشفرة | Character الحرف |
|---|---|
| 032 | (space) |
| 033 | ! |
| 034 | " |
| 035 | # |
| 036 | $ |
| 037 | % |
| 038 | & |
| 039 | ' |
| 040 | ( |
| 041 | ) |
| 042 | * |
| 043 | + |
| 044 | , |
| 045 | - |
| 046 | . |
| 047 | / |
| 048 | 0 |
| 049 | 1 |
| 050 | 2 |
| 051 | 3 |
| 052 | 4 |
| 053 | 5 |
| 054 | 6 |
| 055 | 7 |
| 056 | 8 |
| 057 | 9 |
| 058 | : |
| 059 | ; |
| 060 | < |
| 061 | = |
| 062 | > |
| 063 | ? |
| 064 | @ |
| 065 | A |
| 066 | B |
| 067 | C |
| 068 | D |

| ASCII Value الشفرة | Character الحرف |
|---|---|
| 069 | E |
| 070 | F |
| 071 | G |
| 072 | H |
| 073 | I |
| 074 | J |
| 075 | K |
| 076 | L |
| 077 | M |
| 078 | N |
| 079 | O |
| 080 | P |
| 081 | Q |
| 082 | R |
| 083 | S |
| 084 | T |
| 085 | U |
| 086 | V |
| 087 | W |
| 088 | X |
| 089 | Y |
| 090 | Z |
| 091 | [ |
| 092 | \ |
| 093 | ] |
| 094 | ^ |
| 095 | _ |
| 096 | ` |
| 097 | a |
| 098 | b |
| 099 | c |
| 100 | d |
| 101 | e |
| 102 | f |
| 103 | g |
| 104 | h |
| 105 | i |

| ASCII Value الشفرة | Character الحرف |
|---|---|
| 106 | j |
| 107 | k |
| 108 | l |
| 109 | m |
| 110 | n |
| 111 | o |
| 112 | p |
| 113 | q |
| 114 | r |
| 115 | s |
| 116 | t |
| 117 | u |
| 118 | v |
| 119 | w |
| 120 | x |
| 121 | y |
| 122 | z |
| 123 | ¦ |
| 124 | . |
| 125 | ¦ |
| 126 | ~ |
| 127 | ⌂ |
| 128 | Ç |
| 129 | u |
| 130 | é |
| 131 | â |
| 132 | a |
| 133 | à |
| 134 | a |
| 135 | ç |
| 136 | ê |
| 137 | ë |
| 138 | è |
| 139 | ï |
| 140 | î |
| 141 | ì |
| 142 | A |

| ASCII Value الشفرة | Character الحرف |
|---|---|
| 143 | Å |
| 144 | E |
| 145 | æ |
| 146 | Æ |
| 147 | ô |
| 148 | o |
| 149 | ò |
| 150 | û |
| 151 | ù |
| 152 | y |
| 153 | O |
| 154 | U |
| 155 | ¢ |
| 156 | £ |
| 157 | ¥ |
| 158 | Pt |
| 159 | ƒ |
| 160 | á |
| 161 | í |
| 162 | ó |
| 163 | ú |
| 164 | ñ |
| 165 | Ñ |
| 166 | ª |
| 167 | º |
| 168 | ‹ |
| 169 | ⌐ |
| 170 | ¬ |
| 171 | ½ |
| 172 | ¼ |
| 173 | ı |
| 174 | « |
| 175 | » |
| 176 | |
| 177 | ░ |
| 178 | ▒ |
| 179 | │ |

| ASCII Value الشفرة | Character الحرف |
|---|---|
| 180 | ┤ |
| 181 | ╡ |
| 182 | ╢ |
| 183 | ╖ |
| 184 | ╕ |
| 185 | ╣ |
| 186 | ║ |
| 187 | ╗ |
| 188 | ╝ |
| 189 | ╜ |
| 190 | ╛ |
| 191 | ┐ |
| 192 | └ |
| 193 | ┴ |
| 194 | ┬ |
| 195 | ├ |
| 196 | ─ |
| 197 | ┼ |
| 198 | ╞ |
| 199 | ╟ |
| 200 | ╚ |
| 201 | ╔ |
| 202 | ╩ |
| 203 | ╦ |
| 204 | ╠ |
| 205 | ═ |
| 206 | ╬ |
| 207 | ╧ |
| 208 | ╨ |
| 209 | ╤ |
| 210 | ╥ |
| 211 | ╙ |
| 212 | ╘ |
| 213 | ╒ |
| 214 | ╓ |
| 215 | ╫ |
| 216 | ╪ |
| 217 | ┘ |
| 218 | |
| 219 | █ |
| 220 | ▄ |
| 221 | ▌ |
| 222 | ▐ |
| 223 | ▀ |
| 224 | α |
| 225 | β |
| 226 | Γ |
| 227 | π |
| 228 | Σ |
| 229 | σ |
| 230 | µ |
| 231 | τ |
| 232 | Φ |
| 233 | Θ |
| 234 | Ω |
| 235 | δ |
| 236 | ∞ |
| 237 | Ø |
| 238 | ε |
| 239 | ∩ |
| 240 | ≡ |
| 241 | ± |
| 242 | ≥ |
| 243 | ≤ |
| 244 | ⌠ |
| 245 | ⌡ |
| 246 | ÷ |
| 247 | ≈ |
| 248 | |
| 249 | • |
| 250 | · |
| 251 | √ |
| 252 | n |
| 253 | $^{2}$ |
| 254 | ■ |
| 255 | (blank 'FF') |




## المرجع الأساسى لنظام التشغيل

## MS-DOS 5 / MS-DOS6

يقع هذا الكتاب فى جزئين ، ويشرح جميع اصدارات نظام التشغيل MS-DOS ابتداء من الاصدار 3.1 الى الاصدار 6 ويركز بصفة أساسية على الاصدار ين الخامس والسادس .

يشتمل **الجزء الأول** على الأبواب الرئيسية التالية :

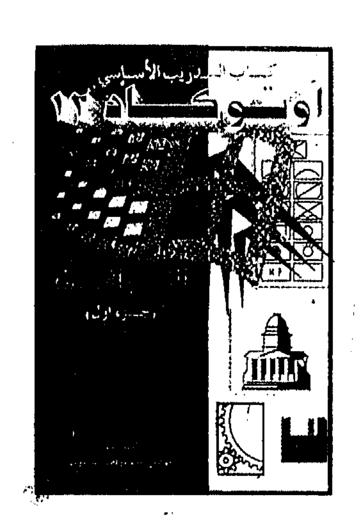
- ◆ تعرف على الحاسب الشخصى
- ◆ دروس عملية للمبتدئين
- ◆ التعامل مع منسق النصوص Editor
- ◆ استخدام "دوس شيل"
- ◆ مرجع شامل للأوامر
- ◆ التعامل مع الملفات والأدلة والأقراص
- ◆ مفاهيم متقدمة تشمل :

* الملفات التجميعية * وصل الأوامر * استخدام برامج ادارة ملحقات الحاسب

* استخدام مفكرة الأوامر والمختزلات * اعادة التوجيه

ويشتمل **الجزء الثانى** على الأبواب الرئيسية التالية :

- ◆ حماية البيانات من الفيروسات باستخدام برنامج MS Anti-Virus
- ◆ النسخ الاحتياطى للملفات ومقارنتها واسترجاعها باستخدام برنامج MS Backup
- ◆ اعادة الملفات الملغية باستخدام برنامج MS Undelete
- ◆ مضاعفة حجم القرص المغناطيسى باستخدام برنامج DoubleSpace
- ◆ زيادة سرعة الحاسب باستخدام برنامج MS Defrag وبرنامج SMARTDRV
- ◆ توفير حجم الذاكرة وتحقيق أقصى استفادة منها باستخدام برنامج MemMaker
- ◆ شرح الأوامر الجديدة فى DOS 6



## كتاب التدريب الأساسى
## أوتوكاد ١٢

يشرح كتاب **التدريب الأساسى أوتوكاد ١٢** الإصدار الأخير من برنامج Autocad الشهير الذى يعمل تحت بيئة التشغيل الرسومية «ويندوز»، ويحتوى على شرح واف لجميع الأوامر اللازمة لإنجاز أى نوع من أنواع الرسومات فى مستوى واحد بين المحورين الأفقى والرأسى، ويستخدم الكتاب مفهوم خطوة..خطوة، ولذلك يمكن أن نعتبره دليل تعليمى يصلح لمن يرغبون فى التعلم الذاتى ومراكز التدريب المتخصصة. والكتاب مفيد لطلاب العلم فى الكليات الهندسية الذين يدرسون علم التصميم بمساعدة الحاسب الآلى. والمهندسين الذين يعملون فى مجال الرسم الهندسى. والفنانين والرسامين والمصممين الذين يهمهم تنسيق الخطوط والدوائر والأشكال الهندسية.

## المرجع الأساسى
## لقاعدة البيانات
## dBASE III PLUS

يشرح هذا الكتاب كيفية استخدام قاعدة البيانات dBASE III PLUS سواء من ناحية الأوامر واستخدام شاشات المساعدة ، أو من ناحية البرمجة والكتاب صيغ بأسلوب تعليمى منظم يصلح للتدريس فى الجامعات والمعاهد العملية ، وسهل ليستفيد منه العاملون فى مجال الحاسبات والمبتدئون على حد سواء .

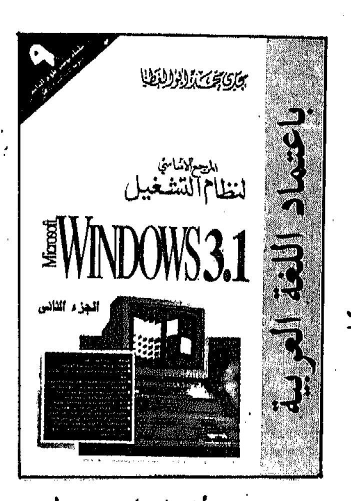
يقع الكتاب فى جزئين . يشرح الجزء الأول كيفية بناء قواعد البيانات واعداد شاشات الادخال ، وترتيب الملفات واستعراض محتوياتها ، والاستفسار عنها ، واستخراج التقارير والملصقات .

ويشرح الجزء الثانى أساسيات البرمجة عموما ، وكيفية البرمجة بقاعدة البيانات . ولذلك فقد جاء شاملا لكل ماتحتويه المادة ، ولكل مايحتاج اليه العاملون فى هذا المجال .

## تعلم مايكروسوفت اكسل
## Excel 4.0 for Windows
## فى يوم واحد




يشرح هذا الكتاب أساسيات التعامل مع برنامج الجداول الالكترونية Microsoft Excel 4.0 والكتاب يخاطب المبتدئين والذين لايجدون الوقت الكافى لقراءة مراجع مطولة مثل كتابنا **المرجع الأساسى لمستخدمى اكسل** وتتلخص فكرته فى تقديم المادة بسهولة وسرعة تتناسب مع الهدف من استخدام Windows ، وهو السهولة والسرعة . والكتاب يبدأ من انشاء صفحة البيانات الالكترونية (Worksheet) وادخال بياناتها وتعديل محتوياتها ، وطباعتها باختيارات متعددة ، الى ادخال تحسينات عليها تساعد فى اظهارها بشكل جيد مثل تغيير أنماط الكتابة واستخدام البراواير والألوان والأنساق المناسبة . وينتهى بالتعامل مع أكثر من صفحة بيانات وتبادل المعلومات بينها وانشاء علاقات دائمة بين المستندات . ثم يشرح كيفية تمثيل البيانات بالرسم البيانى وكيفية حفظ الرسم وطباعته وحذفه واسترجاعة وادخال تحسينات عليه .




# المرجع الأساسي
# لنظام التشغيل
# Microsoft Windows 3.1
# باعتماد اللغة العربية

يشرح هذا الكتاب بيئة التشغيل الرسومية الأكثر راحة والأسرع تفاعلا والمعروفة باسم Microsift Windows أو نوافذ مايكروسوفت ، ويشتمل على قواعد عامة للتعامل مع البرامج التى تأتى ضمن حزمة النوافذ ، أو المصممة للعمل تحت نظام Windows . والكتاب عبارة عن دروس عملية تنهج سياسة "**خطوة .. خطوة** " فى التعليم والتعلم تساعدك فى أن تعلم نفسك :

• التعامل مع نوافذ البرامج بفتحها وغلقها وتحجيمها وترصيصها وتكديسها وترتيب أيقوناتها من سطح المكتب

• تجميع البرامج والمستندات فى نوافذ جماعية واستخدام مدير البرامج لانشاء النوافذ الجماعية ، واعادة ترتيبها ، وتغيير أسمائها ، والغائها ، وتخصيص أيقونات للبرامج .

• تبادل المعلومات بين البرامج المصممة للعمل تحت نظام Windows أو بينها وبين البرامج المصممة للعمل تحت نظام DOS .

• التعامل مع الملفات والأدلة باستخدام مدير الملفات بدلا من استخدام الأوامر الصعبة من محث DOS لفتحها ، أو انشائها ، أو لاظهار محتوياتها ، أو لنقلها أونسخها أو البحث عنها ، أو حذفها أو تغيير أسمائها .

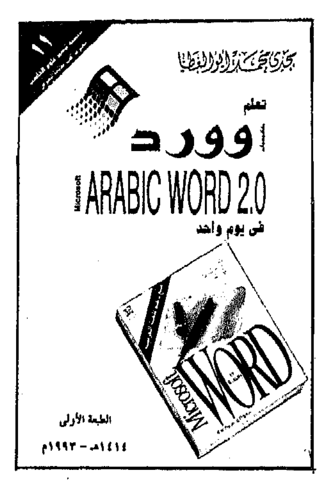
• استخدام لوحة التحكم لتهيئة وضبط النظام ليوافق استخداماتك وحاجاتك الخاصة عن طريق التوصيل الى شبكات الطباعة ، واعداد منافذ الاتصالات ، وضبط الوقت والتاريخ ، وتركيب الخطوط ، واختيار الألوان المناسبة لتجميل منظر "المكتب" .

• استخدام مدير الطباعة لمعاينة طوابير الطباعة وتغيير أولوياتها ، وجر والقاء المستندات للطباعة .

• البرامج المكتبية التى تأتى ضمن حزمة Windows والتى يطلق عليها Accessories وتشمل: الكاتب العربى ، الطرفية ، المفكرة العربية ، التقويم ، الفرشاة ، والساعة ، والحاسبة

• التعامل مع البرامج المصممة للعمل تحت DOS والتحكم فيها بتشغيلها داخل نوافذ أو على شاشة كاملة، وتحجيمها ، والانتقال من برنامج مفتوح لآخر ، وتبادل المعلومات بينها ، وكيفية انشاء وتعديل ملف PIF ليتم تشغيلها تلقائيا بنظام Windows

لذلك فان هذا الكتاب يعتبر بحق أول كتاب يشرح نظام نوافذ مايكروسوفت المدعم للغة العربية شرحا وافيا . بل هو الكتاب الوحيد الذى يشرح تعريب برامج النوافذ والبرامج المكتبية التى تأتى معها




## تعلم مايكروسوفت وورد
## Arabic Word for Windows
## فى يوم واحد

يشرح هذا الكتاب أساسيات التعامل مع برنامج Microsoft Arabic Word والكتاب يخاطب المبتدئين والذين لايجدون الوقت الكافى لقراءة مراجع مطولة مثل كتابنا **المرجـــع الأساسى لمستخدمى وورد** وتتلخص فكرته فى تقديم المادة بسهولة وسرعة تتناسب مع الهدف من استخدام Windows ، وهو السهولة والسرعة . والكتاب يبدأ من انشاء المستند وتعديل محتوياته ، وطباعته باختيارات متعددة ، الى ادخال تحسينات عليه تساعد فى اظهاره بشكل جيد مثل تغيير أنماط الكتابة واستخدام البراواير والألوان والأنساق المناسبة . وينتهى بشرح مفاهيم متقدمة مثل استخدام الأنماط وانشاء الجداول والدمج البريدى



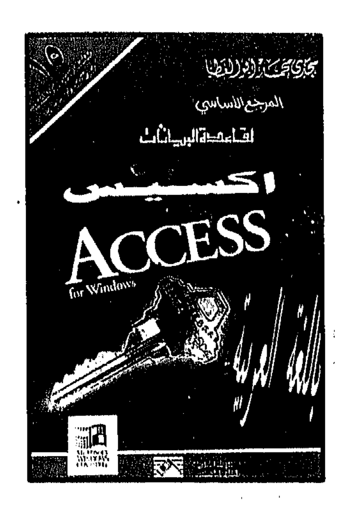

## تعرف على الحاسب الشخصى

يشرح هذا الكتاب كل مايهم القارىء عن تكنولوجيا الحاسب الشخصى وبرامجه وأجهزته ويحتوى على مقدمة لنظم تشغيل الحاسبات بصفة عامة ونظام التشغيل MS-DOS 5. بصفة خاصة ، ومقدمة لنظام الشغيل Microsoft Windows 3.11 المدعم للغة العربية . ويشتمل على دروس عملية لمن يستخدمون الحاسب لأول مرة . ويشرح لمن يخططون لشراء حاسب شخصى أو لتطوير حاسباتهم كيفية ترشيد قرار شراء الحاسب وملحقاته . وللمهتمين بتعريب الحاسبات كيف يختارون تعريب حاسباتهم . وأخيرا مقدمة عن تشبيك الحاسبات .

لذلك فاننا نعتبر أن دراسة هذا الكتاب ليست ضرورية للمبتدئين فقط ، ولكن أيضا لفئات كثيرة تشمل :

* رجال الأعمال والمديرون المهتمون بمكننة أعمالهم ، والذين يخططون لشراء حاسبات شخصية
* طلاب المدارس والمعلمون فى جميع مراحل التعليم الذين ينهجون سياسة "**خطوة .. خطوة** " فى التعليم والتعلم
* الأشخاص الذين يستخمون الحاسب الا أن معلوماتهم عن الأجهزة والبرامج غير كافية

# المرجع الأساسى
# لقاعدة البيانات Access
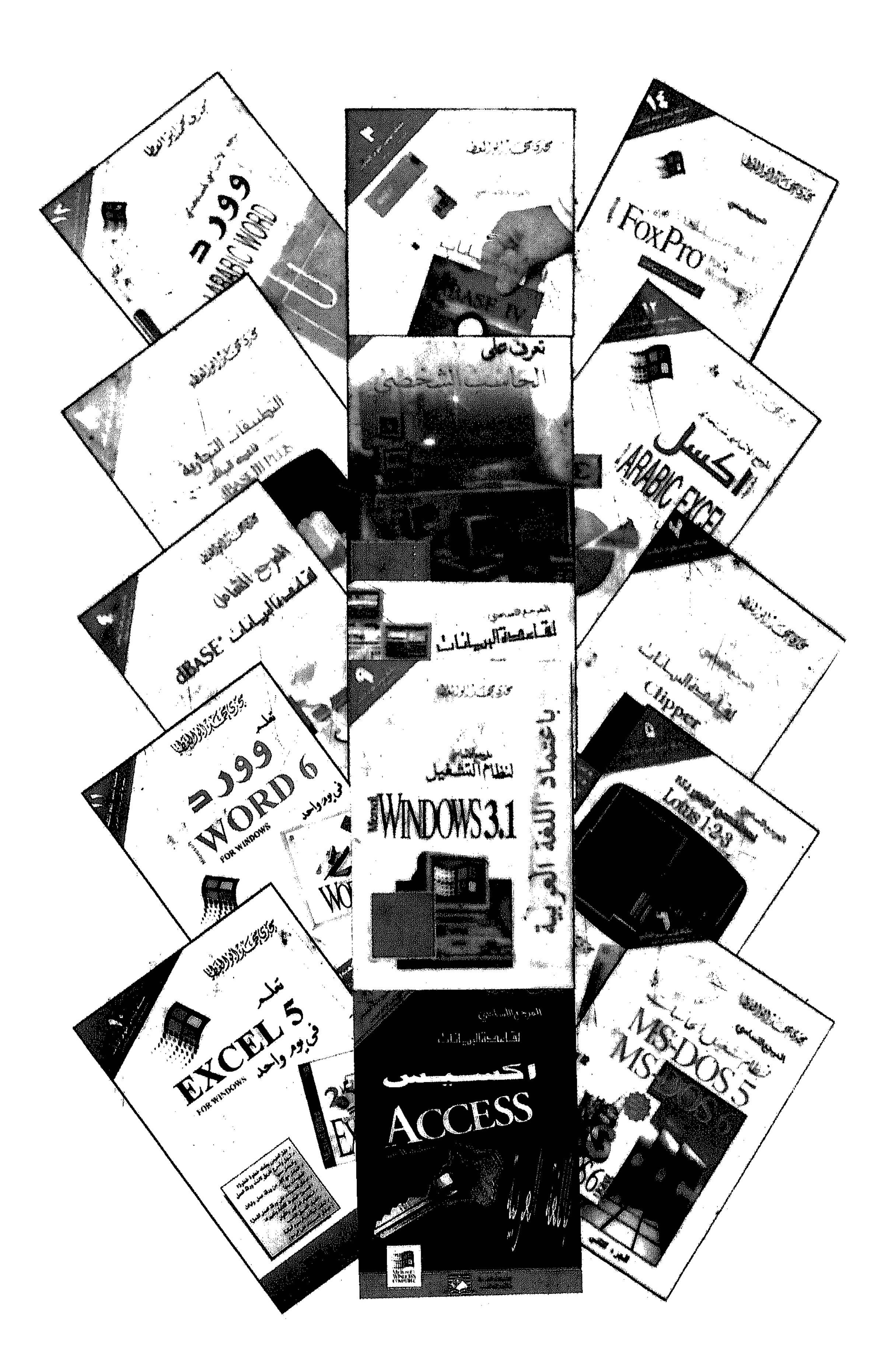



يصلح كتاب المرجع الأساسى لقاعدة البيانات Access 2 لكل من المدربين والمتدربين والمعاهد المتخصصة ، لأنه يعتمد سياسة خطوة خطوة فى التعليم والتعلم ، من خلال تمارين عملية مسجلة على قرص مغناطيسى مرفق مع الكتاب ويباع مجانا. يتكون كل تمرين من خطوات مسلسلة تشتمل على الإجراءات المطلوبة للوصول إلى الهدف ، وفى حالة الضرورة تظهر الشاشات التى توضح نتيجة الإجراء المتخذ داخل التمرين ، والهدف من ذلك تجنب الوقوع فى أى خطأ أثناء تنفيذ الخطوات التالية.

يخاطب هذا الكتاب المبتدئين ومن يستخدمون برنامج Access 2 ، وإذا كانت لك خبرة سابقة بالتعامل مع البرنامج ، فإن الكتاب سيضع يدك على مفاهيم متقدمة وعلى مواضع قوة البرنامج التى تبحث عنها والتى تجعلك تفضل البرنامج على غيره من برامج قواعد البيانات.

يبدأ الكتاب بإعطاء خلفية ضرورية يجب أن تفهمها جيدا قبل أن تبدأ استخدام "أكسس" ثم يشرح الوظائف الأساسية لنظم إدارة قواعد البيانات والتى يحتاجها معظم الناس ومنها :

- انشاء قاعد البيانات وتعديلها.
- اظهار البيانات والتحكم فيها.
- انشاء ملف الاستعلام واستخدامه.
- تصميم التقارير وطباعتها.
- تصميم النماذج واستخدامها.
- ربط الملفات.
- التعامل مع برامج أخرى.
- استخدام الماكروز.
- استخدام اكسس داخل شبكة اتصالات.



وورد
ARABIC WORD
dBASE IV
FoxPro
تعرف على
التطبيقات التجارية
اكسل
ARABIC EXCEL
المرجع الشامل
لقاعدة البيانات
dBASE
لقاعدة البيانات
Clipper
تعلم
وورد
WORD 6
FOR WINDOWS
في يوم واحد
لنظام التشغيل
WINDOWS 3.1
باعتماد اللغة العربية
Lotus 1-2-3
تعلم
EXCEL 5
FOR WINDOWS
في يوم واحد
المرجع الأساسي
لقاعدة البيانات
اكسيس
ACCESS
MS-DOS 5